НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
“ 4 ” грудня 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121 «Інженерія програмного забезпечення»
(код і назва спеціальності)

на тему: Засоби розроблення паралельних програм для метеорологічного прогнозування

Виконав: студент 6 курсу, групи ІТ-73мп
(шифр групи)

Томишин Юрій Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор, д.ф.-м.н., проф., Дорошенко А.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121 «Інженерія програмного забезпечення»
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ролік О.І.
(підпис) (ініціали, прізвище)
« 29 » жовтня 2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Томишину Юрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Засоби розроблення паралельних програм для метеорологічного прогнозування

науковий керівник дисертації професор, д.ф.-м.н., проф., Дорошенко А.Ю. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації 4 грудня 2018 року

3. Об'єкт дослідження паралельне програмування

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) застосуванням засобів автоматизованого конструювання паралельного коду

5. Перелік завдань, які потрібно розробити розв'язання задачі конвективної дифузії, застосування алгебро-алгоритмічного інструментарію для проектування та генерації коду, провести експерименти виконання на багатоядерній

платформі Intel Xeon Phi 7210

6. Орієнтовний перелік ілюстративного (графічного) матеріалу _____

7. Орієнтовний перелік публікацій _____

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 29 жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання теми магістерської дисертації	29.10.2018	
2	Отримання завдання	01.11.2018	
3	Ознайомлення з літературними джерелами, веб-джерелами згідно завдань	05.11.2018	
4	Розробка теоретичної частини	07.11.2018	
5	Узгодження з керівником технічних особливостей реалізації ПЗ	08.11.2018	
6	Узгодження з керівником програмного продукту	09.11.2018	
7	Розробка програмного продукту	12.11.2018	
8	Проведення експериментів	17.11.2018	
9	Написанння тез на конференцію	20.11.2018	
10	Оформлення пояснювальної записки	01.12.2018	

Студент _____ Томишин Ю.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Дорошенко А.Ю.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Виконане автоматизоване конструювання високорівневих алгебро-алгоритмічних специфікацій програмного забезпечення для розв’язання задачі метеорологічного прогнозування. Виконана генерація програмного коду за побудованими специфікаціями на основі використання розроблених інструментальних засобів автоматизованого проектування та синтезу програм. Проведено експеримент з виконання згенерованої паралельної програми метеорологічного прогнозування на багатоядерній платформі Intel Xeon Phi.

Загальний обсяг роботи: 82 с., 5 рис., 29 табл., 30 джерел.

Ключові слова: автоматизоване проектування програм, алгебра алгоритмів, задача конвективної дифузії, метеорологічне прогнозування, паралельні обчислення, синтез програм.

ABSTRACT

The computer-aided design of high-level algebra-algorithmic specifications of the software for solving the problem of meteorological forecasting is performed. Generation of the program code behind the constructed specifications on the basis of use of the developed tools of computer-aided design and synthesis of programs is executed. An experiment was carried out to implement the generated parallel program of meteorological forecasting on the Intel Xeon Phi multicore platform.

Total amount of work: 82 p., pic 5, 29 tables., 30 links.

Keywords: automated software design, algorithm algebra, convection-diffusion problem, meteorological forecasting, parallel computation, software synthesis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

TermWare — система символьных вычислений для разработки динамических приложений на основе правил переписывания.

SMP — симетричне мультипроцесування (англ. Symmetric Multiprocessing).

САА — системи алгоритмічних алгебр.

САА-М — модифіковані системи алгоритмічних алгебр.

ЧПП — чисельний прогноз погоди.

ІПС — інтегрований інструментарій проектування та синтезу програм.

МАУМ — модифікований адитивно-усереднений метод розщеплення.

ВСТУП

Системи прогнозування з року в рік ускладнюються, а обсяг обчислювальних даних стрімко зростає, удосконалюються математичні моделі, які вони застосовують. Завдання метеорологічного прогнозування не є винятком. З огляду в різних галузях людської діяльності, великий попит на метеорологічні прогнози, проектування й розробка програм для вирішення цієї проблеми надзвичайно актуальна. Натомість того, велике, а іноді й вирішальне значення мають надійність і своєчасність метеорологічних прогнозних даних, високі вимоги які пред'являє до швидкісних характеристик таких програмних продуктів.

Завдання метеорологічного прогнозування являє собою складну прикладну обчислювальну задачу зі строгими часовими обмеженнями і високими вимогами до точності вирахуваних результатів. На великих наборах даних присутній великий обсяг обчислень що вимагає розроблення паралельних реалізацій й оптимізації.

В області проектування й дослідження паралельних та розподілених обчислень одним з перспективних напрямків в даний час є створення програмних абстракцій у вигляді алгебро-алгоритмічної мови й моделі, яка спрямована на розвиток архітектурних і мовнонезалежних способів програмування для багатопроцесорних обчислювальних систем і мереж. Запропоновано теорію в працях [12–19], методологію та інструментарій автоматизованого розроблення паралельних програм на основі інструмента високорівневої алгебро-алгоритмічної формалізації та автоматизації програмних перетворень. Інструментальна система автоматизації програмування, була спроектована під назвою Інтегрованим інструментарієм Проектування та Синтезу програм (ІПС) [12-16]. Три форми представлення алгоритмічних знань про предметну область система застосовує: аналітичну (формули в алгебрі алгоритмів), природно-лінгвістичну (текст) і графову (граф-схема). В основі ІПС лежить метод діалогового побудови синтаксично коректних програм, спрямованих на усунення з’явлення синтаксичних помилок в процедурі розроблення. Поетапне розроблення програм забезпечує даний інструментарій, починаючи від формальної класифікації та закінчуючи кодом цільової мови

програмування (Java, C++). Розглядається нова версія системи синтезу програм в працях [17-19] — Онлайновий Діалоговий конструктор Синтаксично Правильних програм (ОД-СП). Відмінна риса інструментів ОДСП є застосування Web-технологій та архітектури розподілених систем.

Використовувалися системи ІПС та ОДСП в попередніх працях [12-19] для розроблення паралельних програм для багатоядерних процесорів і графічних прискорювачів, а також розподілених і сервісно-орієнтованих додатків для Грід. Спроектована інтеграція ІПС й систем переписувальних правил Term-are [15, 20] для автоматизації перетворення паралельних програм. Зокрема, задача метеорологічного прогнозування є предметною областю використання інструментарію. Здійснено автоматизоване розроблення паралельного застосування в праці [18] для розв'язання двовимірної задачі конвективної дифузії, що з'являється в математичній моделі атмосферної циркуляції в метеорології. У даній праці представлені результати використання інструментарію для генерації паралельного застосування для тривимірного випадку зазначеної задачі [21-23]. Розроблення багатопотокового застосування яке реалізоване мовою програмування C++ і застосовує технологію OpenMP [24].

Запропонований у даній праці є близький до праць, пов'язаних з алгебраїчним програмуванням [25], синтезом програм на основі високорівневих класифікацій [26, 27], а також автоматизованою генерацією застосувань OpenMP [18-21] і Java-додатків. Застосування специфікацій алгебри Глушкова є відмінністю нашого підходу, поданої в природно-лінгвістичній формі, що спрощує розуміння алгоритмів і здобуття необхідного рівня якості застосувань. Застосування методу діалогової побудови синтаксично коректних програм є ще однією перевагою спроектованих засобів, які виключають ймовірність виникнення синтаксичних помилок у процесі розроблення схем.

У даній роботі наведено результати використання алгебро-алгоритмічного інструментарію для проектування та проведено експеримент з виконання згенерованої паралельної програми на багатоядерній платформі.

1 ОГЛЯД ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд предментої області

Метеорологія — це наука яка вивчає властивості земної атмосфери її будову та складу, фізичні та хімічні процеси, які в ній відбуваються. Метеорологію ємко і коротко кажучи називають фізикою атмосфери, що найбільшою мірою відповідає її сьогоднішньому розумінню. Метеорологія входить в частину більш загальної науки — геофізика, яка досліджує явища й процеси, що виникають в атмосфері, на плоскості суші та в товщі ґрунтів. Велика доля метеорологів займаються моделюванням прогнозу погоди, клімату, дослідженням атмосфери за допомогою чисельної моделі прогнозу погоди.

Головні завдання метеорології [9]:

- вивчення будови та складу земної атмосфери;
- дослідження теплообігу і теплового режиму в атмосфері та на земній плоскості;
- вивчення всіх фізичних і хімічних процесів та явищ, що здійснюються в атмосфері;
- вивчення волого обігу і фазової перебудови води в атмосфері у зв'язку із земною поверхнею;
- дослідження тутешніх циркуляцій та частин її механізму, атмосферних рухів загальної циркуляції атмосфери;
- дослідження атмосферного електричного поля;
- дослідження в атмосфері оптичних й акустичних явищ;
- розроблення способів управління процесами, які виконуються в земній атмосфері;
- створення фізико-математичних теорій атмосферних процесів, що мають завершальною ціллю прогнозу атмосферних явищ.

1.2 Огляд існуючих рішень

Чисельна модель прогнозування погоди або чисельний прогноз погоди (ЧПП)— комп'ютерна програма, побудована на основі фізичної системи рівнянь і полягає на базі теперішніх даних метеорологічного прогнозу. Дана модель може бути глобальною, що покриває всю поверхню Землі, або локальною, що покриває окрему ділянку поверхні нашої планети.

На теперішній час на нашій планеті всього-навсього існують дві головні моделі прогнозування погоди: американська модель Global Forecasting System (GFS), британська модель European Center for Medium Range Forecasting (ECMWF). Всі інші погодні сервіси, так чи інакше, застосовують відомості розраховані на основі цих моделей або їхніх модифікацій.

1.2.1 Модель прогнозування погоди GFS

Global Forecast System (GFS) — глобальна чисельна модель прогнозу погоди, розробленою і керованою Національною управлінням океанічних і атмосферних досліджень США. Перша версія GFS з’явилася у 1988 році в Національному центрі передбачення стану екологічного середовища США в рамках Глобальної системи асиміляції даних та ціллю поліпшення глобального прогнозу та прогнозу довгохвильових синоптичних систем [10].

GFS — спектральна гідростатична модель, остання версія якої (липень 2010) має 91 рівень по вертикалі й застосовує сігма-систему координат. Горизонтальний крок становить приблизно 27 км, а передчасність прогнозу — 384 години, з яких перші 192 години високої часової дискретизації.

Не зважаючи на поважний вік (що стосується оперативної моделі), GFS була і залишається однією із двох найкращих глобальних моделей прогнозу погоди у світі (іншою є європейська ECMWF Integrated Forecast System). Модель постійно розвивається, збільшується число прогнозованих метеорологічних величин, підвищується якість самих прогнозів, вдосконалюються її експлуатаційні

характеристики та продуктивність. В даний час в Сполучених Штатах Америки діє ряд варіантів цієї моделі, що розрізняються між собою роздільною здатністю та фізичними схемами.

GFS є єдиною глобальною прогностичною моделлю у світі, більшість вихідних даних якої є вільно доступні в інтернеті та складають основу для діяльності численних недержавних компаній, що займаються прогнозуванням погоди.

Переваги моделі GFS:

- доступність прогнозів так як вони надаються абсолютно безкоштовно, будь-хто може завантажити їх на офіційному сайті;
- прогноз погоди на 10, 16 та 30 днів;
- регулярний розрахунок прогнозу погоди для всієї поверхні земної кулі в декількох гідрометеоцентрах в різних країнах;
- прийнятно-точне передбачення погоди для відкритих океанів;
- модель запускається протягом доби чотири рази, робить вирахування три години, запускається в 0, 6, 12 та 18 годин по UTC. Результати публікує через 3.5 години. Так як вона запускається частіше, тому може швидше коригувати свої помилки;
- крок 0.5 градуса (близько 50 км), кожна модель ділить земну кулю на сітку. Чим менше в ній клітинки, тим точніший прогноз, але тим складніше його вирахувати.

Недоліки моделі GFS:

- усереднена погода для квадрата 0,25°, а в багатьох місцях і 0,5°;
- абсолютно непридатність прогнозу для місць поблизу суші (не враховує рельєфу суші, наявність невеликих островів, обриси берегової лінії материків та великих островів), а так само закритих водойм, таких як, наприклад, Середземне море, і вже точно дає абсолютно випадковий результат для річок, озер та закритих бухт.

1.2.2 Модель прогнозування погоди ECMWF

Європейський центр середньострокових прогнозів погоди (англ. European Centre for Medium-Range Weather Forecasts, ECMWF) є незалежною міжурядовою організацією, яка підтримується 34 країнами. ECMWF є одночасно науково-дослідним інститутом та цілодобовою оперативною службою, яка готує та поширює численні прогнози погоди для своїх країн-учасниць. Ці дані повністю доступні для національних метеорологічних служб для країн-учасниць. Центр також пропонує каталог прогнозних даних, які можуть бути придбані компаніями по всьому світу та іншими комерційними клієнтами.

Організація була створена в 1975 році й зараз налічує близько 350 осіб з більш ніж 30-ти держав. ECMWF є одним із шести членів координованих організацій, до яких також входять Рада Європи, Європейське космічне агентство, Організація Північноатлантичного договору (НАТО). У місті Редінг знаходиться Штаб-квартира ECMWF (Велика Британія).

Мабуть, найважливішою подією в європейській метеорології за останні півстоліття стало створення ECMWF. Місія цього центру є постійне вдосконалення моделі прогнозування погоди, а також надання середньострокових прогнозів від деякої кількості днів до деякої кількості сезонів вперед. Сьогодні цей центр є світовим лідером в проектуванні та вдосконаленні моделей прогнозування погоди. 1 серпня 1979 року був розрахований перший прогноз погоди в ECMWF.

Переваги моделі ECMWF:

- прогноз погоди на 10, 15 та 30 днів;
- прогноз погоди на 10 днів вперед, виконується двічі на день;
- модель запускається протягом доби два рази в 0 і 12 годин по UTC;
- Крок 0.25 градусів (близько 25 км) кожна модель ділить земну кулю на сітку;
- Прогноз на місяць вперед, виконується раз на тиждень.

Недоліки моделі ECMWF:

- за доступ тільки до власних файлів з даними потрібно платити близько 150000 доларів на рік;
- краще підходить для середніх широт Європи.

1.3 Підстава для розробки

Нагальна необхідність підвищення продуктивності програмного забезпечення для вирішення трудомістких завдань, з одного боку, і нові можливості розпаралелювання обчислень, що надаються багатоядерною архітектурою (multicore та manycore) сучасних мікропроцесорів з іншого, стимулює створення спеціалізованих інструментів для розробки паралельних програм для таких архітектур. Розробка ефективних програм для багатопроцесорних платформ є масштабна науково-технічна проблема, успішне розв'язання якої може бути забезпечене лише спеціалізацією в предметних областях й глибоким охопленням етапів життєвого циклу програм, що розробляються з використанням засобів автоматизації проектування і програмування, від написання вихідних специфікацій до генерації виконуваного коду. Основою такої автоматизації є, перш за все, високорівнева формалізація проектування багатопоточних програм і автоматизація формальних перетворень програм з метою оптимізації їх роботи.

Прогнозування погоди — складне прикладне обчислювальне завдання з високими вимогами до точності результатів і жорсткими тимчасовими обмеженнями. Великий обсяг обчислень на великих наборах даних вимагає запровадження паралельних реалізацій. У даній роботі будуть представлені результати застосування алгебро-алгоритмічного інструментарію для проектування та генерації коду паралельної програми для розв'язання задачі конвективної дифузії, що виникає при математичному моделюванні атмосферної циркуляції в метеорології. Буде проведено експеримент з виконання згенерованої паралельної програми на багатоядерній платформі Intel Xeon Phi 7210.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Функціональні вимоги

До розроблюваної паралельної програми для для метеорологічного прогнозування були виділені наступні функціональні вимоги:

- завантаження даних з файлу;
- збереження даних в файл;
- відлік часу;
- вивід результатів;
- розв'язання задачі конвективної дифузії.

2.2 Нефункціональні вимоги

До розроблюваної паралельної програми для для метеорологічного прогнозування були виділені наступні нефункціональні вимоги:

- застосування алгебро-алгоритмічного інструментарію для проектування та генерації коду;
- мова програмування C++;
- технологія розпаралелення OpenMP;
- виконання на багатоядерній платформі Intel Xeon Phi 7210.

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

В якості основи функціональної схеми використовується діаграма варіантів використання.

3.1 Діаграма варіантів використання

В додатку Б приведена діаграма варіантів використання системи для метеорологічного прогнозування. У даній системі можна виділити наступний суб'єкт та відповідні прецеденти: Користувач який може запустити обчислення або додати файл з даними для обчислення, для цього прецеденту встановлено відношення включення прецеденту «Перевірка на коректність». До прецеденту «Запустити обчислення» встановлено відношення включення наступних прецедентів:

- завантаження даних із файлів;
- обчислення задачі 3-вимірної конвективної дифузії;
- збереження даних у файли;
- вивід результатів.

3.2 Детальний опис кожного прецеденту

Даний розділ включає детальний опис всіх прецедентів які подано в таблицях 3.1 – 3.7.

Таблиця 3.1 — Описова специфікація прецеденту «Додати файли»

Розділ	Опис
Короткий опис	Для того щоб провести обчислення потрібно додати вхідні дані у вигляді текстового файлу який має бути визначений за встановленими критеріями
Суб'єкт	Коритувач
Передумови	Путь до файлу правильний, файл можна завантажити
Основний потік	Користувач додає текстовий файл з даними, потім він проходить перевірку на коректність

Продовження таблиці

Розділ	Опис
Альтернативний потік	—
Постумови	Файл успішно завантажений

Таблиця 3.2 — Описова специфікація прецеденту «Перевірка на коректність»

Розділ	Опис
Короткий опис	Дані в завантаженому текстовому файлі повинні відповідати встановленим вимогам для того щоб приступити до обчислень
Суб'єкт	Коритувач
Передумови	Дані в файлах коректні, вони виконані за встановленими критеріями
Основний потік	Після того як користувач додав текстовий файл з даними, файл відправляється на перевірку на коректність
Альтернативний потік	—
Постумови	Файл завантажений і він відповідає встановленим вимогам, користувач може приступати до обчислень

Таблиця 3.3 — Описова специфікація прецеденту «Запуск обчислення»

Розділ	Опис
Короткий опис	Після того коли користувач завантажив файл з коректними даними стає можливим обчислення їх
Суб'єкт	Коритувач
Передумови	Наявність коректних даних
Основний потік	Запуск обчислення приводить до поетапного його виконання
Альтернативний потік	—
Постумови	Запуск обчислень запущено

Таблиця 3.4 — Описова специфікація прецеденту «Завантаження даних із файлу»

Розділ	Опис
Короткий опис	Завантаження даних із файлу відбувається шляхом видобування даних із файлу в масиви обчислювальної програми
Суб'єкт	Коритувач
Передумови	Наявність коректних даних
Основний потік	Видобування даних із файлу в масиви обчислювальної програми
Альтернативний потік	—
Постумови	Дані з текстового файлу завантажені в масиви обчислювальної програми

Таблиця 3.5 — Описова специфікація прецеденту «Обчислення задачі 3-вимірної конвективної дифузії»

Розділ	Опис
Короткий опис	Обчислення задачі 3-вимірної конвективної дифузії для метеорологічних прогнозів
Суб'єкт	Коритувач
Передумови	Дані завантажені в масиви обчислювальної програми
Основний потік	Проводиться обчислення даних
Альтернативний потік	—
Постумови	Обчислення закінчено

Таблиця 3.6 — Описова специфікація прецеденту «Збереження даних у файли»

Розділ	Опис
Короткий опис	Після обчилень їх результат запусується в файли
Суб'єкт	Коритувач
Передумови	Сформований новий файл
Основний потік	Завантаження даних в новий сформований файл
Альтернативний потік	—
Постумови	Збереження даних закінчено

Таблиця 3.7 — Описова специфікація прецеденту «Вивід результату»

Розділ	Опис
Короткий опис	Після закінчкння всіх етапів обчислення виводиться результат
Суб'єкт	Коритувач
Передумови	Всі етапи обчислення закінчено
Основний потік	Вивід результату
Альтернативний потік	—
Постумови	Вивід результату відбувся

В додатку В приведена діаграма послідовності яка відображає потік подій, що відбуваються в рамках використання системи.

4 СТРУКТУРНА СХЕМА СИСТЕМИ

Структурні схеми створюються на початкових етапах розробки й передують розробці схем інших типів. Структурна схема показує головні функціональні частини системи, їх призначення і взаємозв'язки між ними. В найзагальнішому вигляді схема зображує принцип дії системи. Структурна схема паралельної програми представлена в додатку И та розгортання цієї програми в додатку К.

4.1 Розгортання системи

Розгортання системи показує фізичне розташування системи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення. Діаграма розгортання містить наступні складові:

- веб-портал надає користувальницький інтерфейс кінцевому користувачу для доступу до сервісів системи;
- контролер — сполучна ланка між моделлю та представленням, одержує дані від користувача, передає їх сервісу доступу до даних, одержує оброблений підсумок і передає його в представлення;
- представлення описує зовнішній вигляд програми;
- сервіс доступу до даних вміщає так звану «Бізнес-логіку» — обробку й верифікацію даних, звертання до баз даних, представляє внутрішній устрій системи. Він не повинен безпосередньо взаємодіяти з користувачем;
- віджет — спеціальний елемент графічного інтерфейсу, який відображає необхідну інформацію або надає можливість взаємодіяти з операційною системою або додатком;
- браузер — програма, яка дозволяє переглядати вміст веб-сторінокж

- служба управління метеорологічними прогнозами — служба, яка здійснює функції планування, синхронізації та керування системою. Вона отримує й обробляє запити, що мають дані про час, місце й тип прогнозу;
- сервіс розрахунку метеорологічних прогнозів здійснює розрахунок метеорологічних прогнозів з застосуванням паралельної програми, яка виконується на багатопроцесорній платформі;
- база даних системи вміщає результатні дані прогнозів, які вираховуються на базі вихідних даних. Використовується в якості системи управління базами даних Oracle Database;
- постачальниками метеорологічних даних є один або кілька постачальників вихідних метеорологічних даних, які є базою для розрахунку прогнозів.

4.2 Структурна схема засобу

Даний програмний продукт складається з наступних складових:

- ініціалізація даних — присвоєння початкових значень змінним та масивам які зберігають дані метеорологічного прогнозування;
- почати відлік часу — запуск відліку часу за який проміжок часу виконується паралельний алгоритм програми;
- завантажити дані в масиви — з текстового файлу читання метеорологічних вихідних даних які є основою метеорологічного прогнозування;
- визначення кількості паралельних потоків — чисельність потоків яка потрібна для правильної роботи паралельного алгоритму;
- розпаралелення — розподіл виконання програми між паралельними потоками;

- цикл — циклічне виконання поки n (змінна m-циклу) помножена на tau (часовий крок) менше або дорівнює TmLimCalc (кінцеве значення часу в умові m-циклу);
- вибір значення поточного напрямку — вибір обчислення задачі за одним з трьох напрямків (λ — довгота, z — висота над рівнем моря, φ — широта);
- поміняти місцями значення в масивах pR1 та pR2 — означає в масив pR1 вставити значення pR2, а в pR2 вставити значення pR1, для цієї дії використовується проміжний масив;
- синхронізатор — це механізм, який дає можливість забезпечити цілісність якого-небудь ресурсу (файл, дані в пам'яті), коли його використовуються декількома процесів або потоків;
- обчислити середні значення на основі отриманих результатів для кожного напрямку;
- збільшити n на M_prm — до n додати M_prm;
- звільнення пам'яті — видалення проміжних масивів які використовували фізичну пам'ять;
- повернути значення NmbThreads — повернення числа кількості паралельних потоків;
- закінчити відлік часу та вивести результат — зупинити відліку часу за який проміжок часу виконується паралельний алгоритм програми та вивести результат на екран;
- порівняти інтерпольовані та обчислені значення в нормі L2;
- зберегти дані у файли для 3-вимірної задачі конвективної дифузії — отримані дані записуються в текстові файли.

5 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

При розробці системи потрібно проявляти певну стриманість. Одне з важливих рішень, з яким не варто поспішати — вибір технологій. У сьогоднішньому швидко розвиваючому кліматі технологічних інновацій успішний вибір і впровадження правильної технології у відповідності з унікальними потребами та цілями може бути більш складним, ніж коли-небудь. Вибір технологій розробки є одним з головних завдань, що виникає на початковому етапі створення будь-якої системи.

Визначення того, яка технологія забезпечить найкращу довгострокову цінність і задовольнить потреби, вимагає глибоких технічних знань і ретельно певних вимог. І як тільки технологія обрана, вона повинна бути успішно впроваджена для досягнення максимальної віддачі. Потрібно бути об'єктивному в тому, як підходити до даної ситуації, можучи розробити рішення, яке найкращим чином відповідає потребам. Технології розробки повинні оптимально відповідати поставленим завданням, що дозволить уникнути багатьох проблем під час проектування та розробки. В результаті повинні бути обрані технології які відповідають наступним вимогам:

- підходять для вирішення даних потреб;
- простота підтримки;
- добре документові;
- економічні в експлуатації;
- безпечні;
- висока продуктивність;
- надійність.

5.1 Алгебра алгоритмів та алгеброалгоритмічний інструментарій проектування для автоматизованої розробки програм

В основу даного напрямку до розроблення паралельних програм покладений агрегат систем алгоритмічних алгебр (САА) та їх модифікацій [12]. Модифіковані САА (САА-М) розраховані для формалізації операцій мультиобробки, що з'являються при проектуванні програмного забезпечення в мультипроцесорних системах. На САА-М базуються створені інструментальні засоби автоматизованого проектування та генерації програм [12–19].

5.1.1 Головні операції алгебри алгоритмів.

САА-М є двоосновною алгеброю <*Op*, *Pr*; Ω,> , де *Op* – множина операторів; *Pr* – множина логічних умов (предикатів); Ω – сигнатура, що складається з логічних операцій (кон’юнкції, диз'юнкції, заперечення, лівого множення оператора на умову) та операторних операцій (композиції, альтернативи, циклу та ін.). Оператори та предикати можуть бути базисними або складеними [2].

На САА-М базується алгоритмічна мова САА/1 [12], призначена для багаторівневого структурного розроблення і документування послідовних та паралельних алгоритмів та програм. Плюсом її застосуванням є можливість опису алгоритмів у природно-лінгвістичній формі, яка є зручною для людини, що спрощує досягнення необхідного рівня програм. САА-схемами називають представлення операторів мовою САА/1.

Слідом за цим наведено список назв та специфікації головних операторних операцій сигнатури САА-М, поданих у природньо-лінгвістичній формі.

1. Композиція (покрокове виконання операторів):

оператор 1 ПОТІМ *оператор* 2

або

оператор 1; *оператор* 2 [12]

2. Альтернатива (умовний оператор):

ЯКЩО умова ТО

оператор 1

ІНАКШЕ

оператор 2

КІНЕЦЬ ЯКЩО [12]

3. Цикл:

ПОКИ *умова виконання циклу*

ЦИКЛ *оператор*

КІНЕЦЬ ЦИКЛУ [12]

4. Операція виконання одного з n операторів за правильності відповідної умови:

ВИБІР ([*умова 1*] → *оператор 1*,

[*умова 2*] → *оператор 2*,

...

[*умова n*] → *оператор n*) [12]

5. Асинхронна диз'юнкція – паралельне виконання *n* операторів (потоків), де *i* – номер потоку:

ПАРАЛЕЛЬНО (*i* = 0, ..., *n*–1)

(

оператор

) [12]

6. Синхронізатор, виконання обчислень зупиняється доти, поки значення умови не стане істинним:

ЧЕКАТИ *умова* [12]

Приклад використання наведених конструкцій наведено у розділі 6.2.

Зазначимо, що операції САА-М також можуть бути представлені російською мовою або наприклад як у роботі [19] англійською мовою.

5.1.2 Додаткові операції алгебри для об’єктно-орієнтованої парадигми

Сигнатура CAA-M також містить операції, які застосовуються для формалізації основних понять об’єктно-орієнтованого програмування (класів, інтерфейсів та ін.). Зазначимо, що для спільності з мовою програмування Java в сигнатуру присутні операції для встановлення анотацій. Анотації є спеціальною формою синтаксичних метаданих, яка може бути додана у програмний код, і використовується для його аналізу, компіляції або виконання. Анотованими можуть бути пакети, класи, методи, змінні та параметри. Далі за цим наведемо список головних операцій CAA-M для позначення класів, що застосовується у даній праці.

1. Позначення анотованого класу:

Annotated class X implements Y (*Class name*, *Interface name*)

<Annotations>

Annotation (*Annotation text*);

Annotation (*Annotation text*);

...

<Fields>

operators

<Methods>

operators

де *Class name* – назва класу, який імплементує інтерфейс *Interface name*; <Annotations>, <Fields>, <Methods> – строки, після яких треба вказати оператори позначення анотацій, полів даних та методів класу, відповідно до цього; *Annotation text* – текст анотації.

2. Анотації визначаються за допомогою конструкції:

Annotation (*Annotation text*),

де *Annotation text* – параметр, де вказується текст анотації.

Зразком позначення анотації для класу може бути конструкція:

Annotation (*Service*).

3. Ініціалізоване поле даних:

Field initialized (*Modifiers*, *Field type*, *Field name*)

<Initialization>

operator

Дана конструкція застосовується для позначення поля класу, для якого встановлюється певне початкове значення. Тут Modifiers – перелік модифікаторів доступу (наприклад, public, static, final і т. п.); Field type – тип даних поля; Field name – назва поля; <Initialization> – рядок, після якого треба вказати оператор або вираз, значення якого буде використане для ініціалізації поля.

4. Позначення анотованого поля даних:

Field annotated (*Modifiers*, *Field type*, *Field name*)

<Annotations>

Annotation (*Annotation text*);

Annotation (*Annotation text*);

... [12]

Словом annotated відмічене поле, позначене анотаціями (наприклад, *Resource*), які треба вказати після рядка <Annotations>.

5. Позначення анотованого методу:

Method annotated (*Return type*, *Method name*, *Parameters list*),

<Annotations>

Annotation (*Annotation text*);

Annotation (*Annotation text*);

...

<Body>

operators [12]

Де *Return type* – тип значення, яке повертає метод; *Method name* – назва методу; *Parameters list* – перелік формальних параметрів; <Annotations> – рядок, після якого вказуються анотації до методу (наприклад, *Override*, *Transactional*); <Body> – рядок, після якого зазначаються оператори тіла метода.

6. Конструкція для виклику методу екземпляра класу має вигляд:

Call instance method (*Instance name*, *Method name*, *Arguments*),

де *Instance name* – ідентифікатор екземпляра класу; *Method name* – назва методу класу; *Arguments* – перелік фактичних параметрів методу [12].

5.1.3 Алгеброалгоритмічний інструментарій проектування

Алгеброалгоритмічний напрям підкріплений низкою інструментальних засобів, створених у рамках київської кібернетичної школи. Першим таким засобом був МУЛЬТИПРОЦЕСИСТ, розроблений у 80-х рр. у відділі автоматизації програмування Інституту кібернетики імені В.М. Глушкова. Даний інструментальний засіб надає багаторівневе проектування алгоритмів і структур даних, поданих у вигляді САА-схем. Алгебраїчний підхід також застосовується при розробці перетворень схем алгоритмів із використанням системи АНАЛІТИК. Спосіб керуючих просторів, що базується на алгебраїчному підході, застосовувався в технології паралельного програмування ПАРУС.

Становленням вищезгаданих інструментальних засобів є інтегрований інструментарій проектування й синтезу програм (система ІПС), створений в Інституті програмних систем НАН України. На алгебрах алгоритмів та алгебро-граматичних методах генерації програм базується дана система. Система розрахована для автоматизації створення САА-схем алгоритмів та синтезу програм мовами програмування С++, Java та ін. В ІПС САА-схема розроблюється в режимі діалогового конструювання із наданням її синтаксичної коректності, на відміну від системи МУЛЬТИПРОЦЕСИСТ, орієнтованої на синтаксичний аналіз САА-схеми (за участю якої відбувається генерація програми). Застосовуваний в інструментарії метод діалогового конструювання синтаксично правильних програм (ДСП-метод) [2] направлений не на пошук та виправлення помилок (як у класичних синтаксичних аналізаторах), а на відсутність та ймовірність їхньої виникнення у процесі побудови алгоритмів. Відмінна риса даної програмної системи є також інтеграція трьох форм подання алгоритмів (аналітичної, природно-лінгвістичної і графової) [3].

Головна ідея методу складається в по рівневому конструюванні схем зверху вниз підходом суперпозиції мовних конструкцій САА-М. Система в діалозі з користувачем надає на вибір тільки ті конструкції на кожному етапі конструювання, підставлення яких у текст алгоритму, що проектується, не переступає синтаксичну вірність схеми. Автоматична генерація тексту програми цільовою мовою програмування відбувається на базі побудованої схеми алгоритму. Відображення операцій САА-М у текст мовою програмування подане у вигляді шаблонів і зберігається в базі даних інструментарію.

Вказаний аспект був виконаний в системі ІПС, розглянутій у працях [12–16]. Інтегрований інструментарій проектування та синтезу алгоритмів і програм складається з таких головних компонентів (рис. 5.1):

- діалоговий конструктор синтаксично правильних програм (ДСП-конструктор), призначений для діалогового проектування схем алгоритмів та синтезу програм;
- редактор граф-схем алгоритмів;
- база даних алгеброалгоритмічних специфікацій, у якій зберігається текст конструкцій САА і базисних елементів схем, а також їх програмні реалізації;
- генератор САА-схем на основі схем більш високого рівня (гіперсхем).

Для автоматизації здійснення трансформацій алгоритмів система ІПС застосовується разом з системою TermWare [20], що базується на парадигмі переписувальних правил.

Нова версія системи ІПС, названа онлайновим діалоговим конструктором синтаксично правильних програм розглядається в працях [17–19]. Система ОДСП спеціалізована для діалогового проектування, генерації й запуску програм. Головна відмінність інструментарію ОДСП у зіставленні з системою ІПС полягає у сервісно-орієнтованій архітектурі інструментарію та спрямованості на багатокористувальницьке використання інструментарію через Інтернет. Іншою відмінністю є орієнтація системи ОДСП на парадигму об’єктно-орієнтованого програмування, тоді як інструментарій ІПС в основному спрямований на імперативну парадигму.

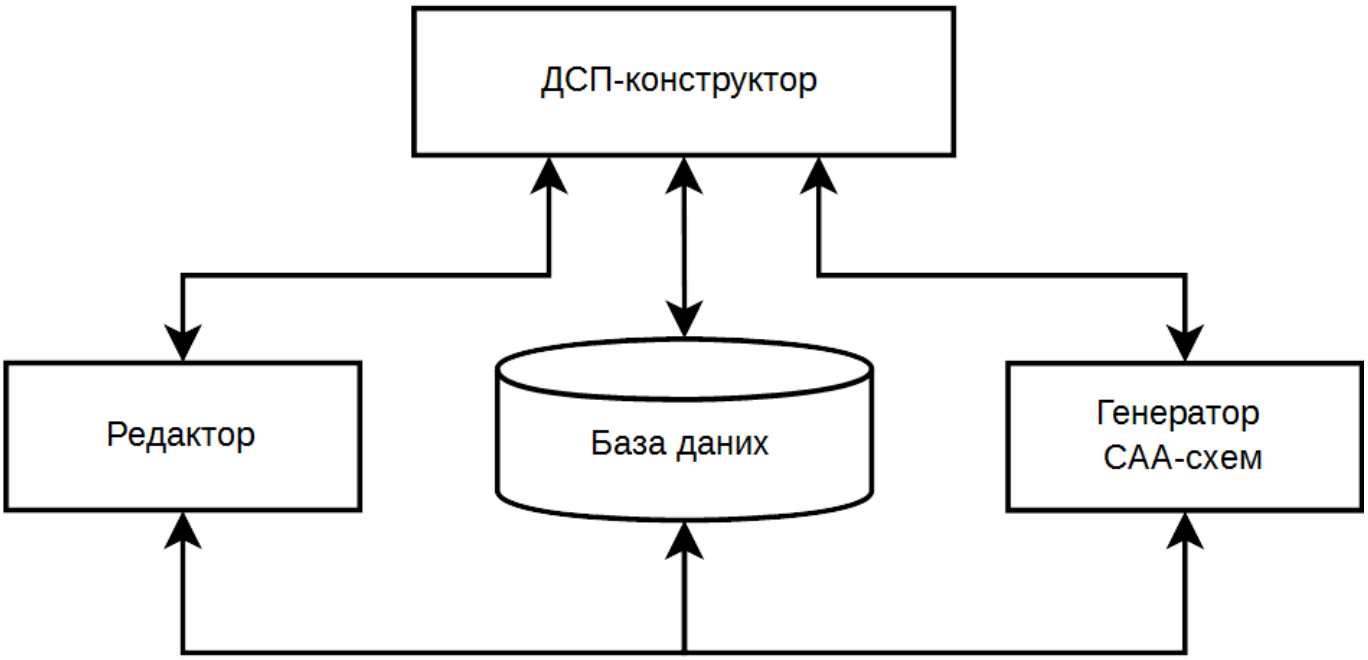

Рисунок 5.1 — Архітектура інтегрованого інструментарію проектування

5.1.3.1 Діалогове конструювання схем алгоритмів

Діалоговий режим з застосуванням меню підстановок і дерева конструювання алгоритму покладено в основу функціонування ДСП-конструктора (рис 5.2). Меню складається з конструкцій САА-М, суперпозиція яких дозволяє створювати алгоритми в згаданих раніше формах. Обрані користувачем конструкції, а також операторні та логічні змінні, що входять в них, відображаються в дереві з подальшою конкретизацією змінних. В залежності від типу вибраної змінної, система пропонує належний перелік операцій САА-М або базисних понять з середовища конструювання алгебро алгоритмічних описів (СКАО). Відмітимо по рівневий стиль конструювання алгоритму, а також можливість переходів на різні рівні з продовженням процесу діалогового конструювання.

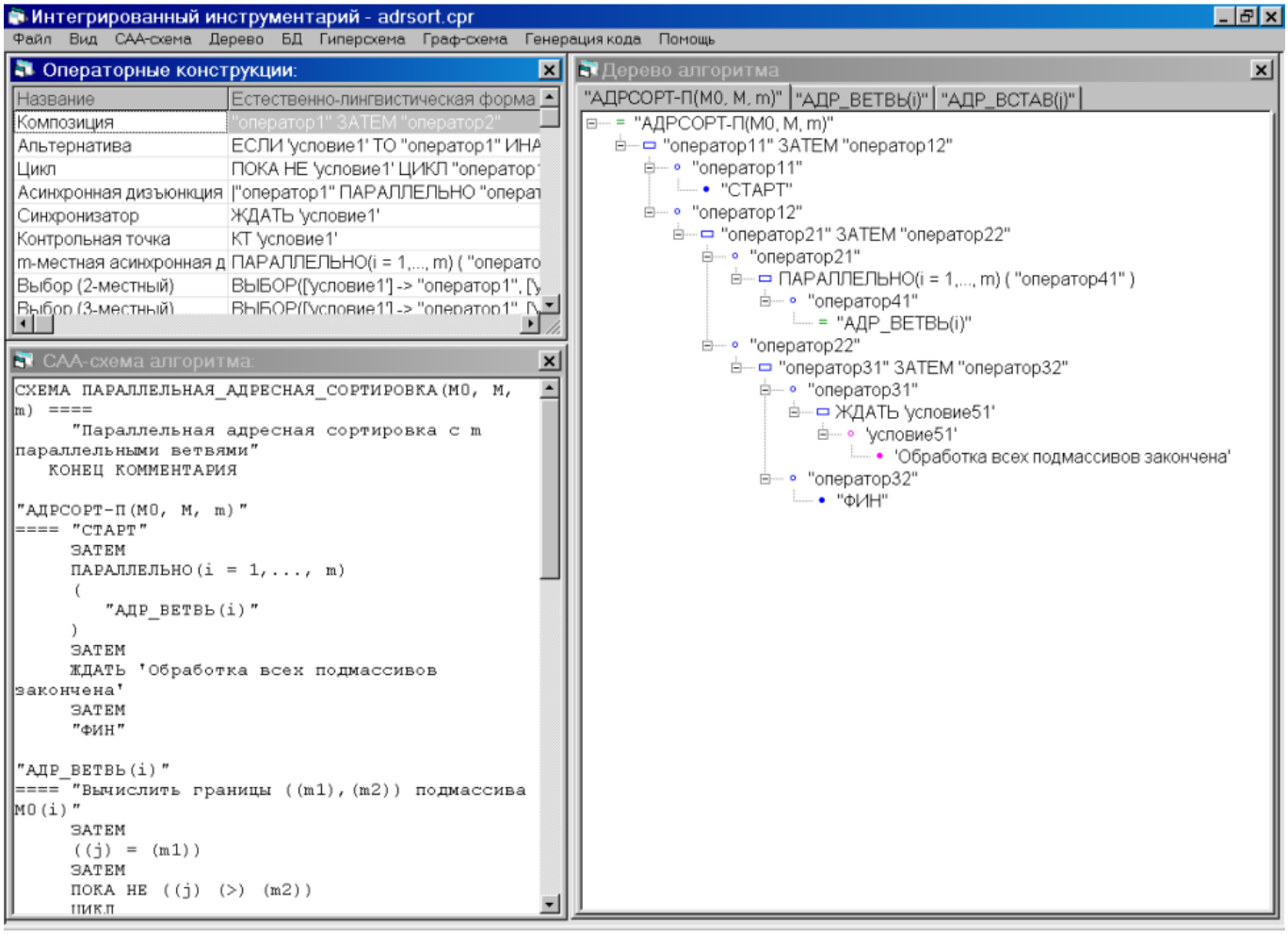

Рисунок 5.2 — Вікно ДСП-конструктора

5.1.3.2 Синтез програм за схемами алгоритмів

Отриманому в ході конструювання алгоритму дерева, реалізаціями елементарних операторів та умов на цільовій об'єктно-орієнтованій мові програмування, а також з іншими фрагментами, ДСП-конструктор виконує синтез програми. У ході синтезу керуючі конструкції САА-схеми алгоритму (асинхронна диз'юнкція, композиція, альтернатива, цикл та ін) відображаються у відповідні оператори мови програмування, а замість базисних операторів і предикатів підставляється їх реалізація на цій же мові з СКАО. Складені оператори можуть бути представлені як підпрограми (методи). На вхід синтезатора надходить також файл, який містить каркасний опис основного класу програми (без реалізацій методів), в який виконується підстановка синтезованого коду.

5.1.3.3 Синтез паралельних багатопотокових програм

Асинхронні алгоритми, представлені в САА-М, в таких мовах програмування, як C++, Java, Perl, Python, Delphi, можуть бути реалізовані за допомогою підпроцесів або потоків. Потік в багатопотокових операційних системах — найменша одиниця виконання. Для кожного процесу (об'єкта, створюваного при запуску програми) ОС створює один головний потік, який є потоком команд центрального процесора, які виконуються по черзі. При необхідності головний потік може створювати інші потоки, користуючись для цього програмним інтерфейсом ОС. Всі потоки, створені процесом, виконуються в одному адресному просторі цього процесу і мають доступ до його ресурсів. Якщо процес створив кілька потоків, то всі вони виконуються паралельно, причому час центрального процесора (або кількох центральних процесорів в мультипроцесорних системах) розподіляється між цими потоками. Кожному потоку дається певний інтервал часу, протягом якого він знаходиться в активному стані.

5.2 Мова програмування

Жодна мова програмування не можна назвати об'єктивно кращою в порівнянні з іншими. Перед початком нового програмного проекту необхідно вирішити, яку мову програмування використовувати. Низькорівневі мови гарні для оптимізації швидкості виконання або розміру програми, в той час як високорівневі мови гарні для створення ясного і добре структурованого коду.

Паралельне програмування стає все більш поширеним явищем в розробці програмного забезпечення із зростанням популярності багатоядерних процесорів. Сьогодні є багато людей та інститутів, які розробляють API паралельного програмування для вже існуючих мов програмування. Новим програмістам важко вибрати мову програмування для паралельної програми, коли кожна мова програмування має API для паралельної реалізації. У цьому розділі було проведено огляд трьох най популярних мов програмування: C++, Java, C#.

5.2.1 Мова C++

C++ — компільована, статично типізована мова програмування для загального призначення. Вона є розширеною мовою C, яка зазвичай використовується для об'єктно-орієнтованого програмування. Вона бере свій початок більше ніж тридцять років тому. Не зважаючи на те, що це далеко не сама стара комп'ютерна мова, це одна з найстаріших, яка широко використовується сьогодні, тому може адаптуватися до сьогоднішнього мінливого технологічного часу.

Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції. Стандартна бібліотека включає, в тому числі, загальновживані контейнери та алгоритми. C++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником — мовою C, найбільшу увагу приділено підтримці об'єктно-орієнтованого та узагальненого програмування.

Мова C++ була розроблена Б'ярном Страуструпом, який зробив першу роботу з розробки в рамках свого PhD-проекту. У перші роки він називав мову «C with classes» («Сі з класами»). Він приступив до розробки нової мови, оскільки, на його думку, жодна з існуючих мов не є ідеальною для великомасштабних проектів. Пізніше, коли він працював в Bell Labs, він знову відчував себе обмеженим. Він обтрусив свій «C with classes» і додав особливості інших мов. Сильний вплив зробила мова Симула, свою роль зіграла мова Алгол. Зрештою, було додано набагато більше, ніж класи: віртуальні функції, шаблони та перевантаження операторів.

C++ виріс далеко за межі розробки однієї людини. Назва насправді походить від іншого розробника, Ріка Маскітті. Це була частково гра на ім'я оператора «++» і частково посилалась на поліпшення (знак плюса означає удосконалення мови і надання їй нового функціоналу).

Мова була вперше стандартизована в 1998 році. Стандарти були знову випущені у 2003, 2007 та 2011 роках. C++ підтримується ISO, великим комітетом по

стандартах. За словами Страуструпа, найбільше покращення полягає в механізмах абстракції. Серед інших цілей останньої редакції: зробити C++ кращою мовою для вбудованих систем і поліпшити підтримку для новачків. Розвиток спирається на певні ідеали. C++ прагне бути переносимим, робиться спроба уникнути залежності від функцій, що залежать від певної платформи.

Стандартні бібліотеки C++ в більшості своїй є надмножиною стандартних бібліотек C. велика частина бібліотеки C++ включає стандартну бібліотеку шаблонів (STL). STL надає такі корисні інструменти, як ітератори (які нагадують високого рівня покажчики) і контейнери (які нагадують масиви, які можуть автоматично зростати за рахунок включення нових елементів). Як і в C, особливістю доступу до бібліотеки відбувається з допомогою директиви #include для підключення стандартних заголовних файлів. C++ надає п'ятдесят не застарілих стандартних заголовків.

При створенні C++ Б'ярн Страуструп ставити за мету:

- отримати універсальну мову зі строгими типами даних, ефективністю та переносимістю мови C;
- безпосередньо і всебічно підтримувати безліч стилів програмування, в тому числі процедурне програмування, абстракцію даних, об'єктно-орієнтоване програмування та узагальнене програмування;
- дати розробнику свободу вибору, навіть якщо це дасть йому можливість вибирати неправильно;
- зробити можливим легкий перехід від програмування на мові C, за рахунок максимального збереження сумісність з мовою C;
- виключивши різновид між C та C++: будь-яка конструкція, допустима в обох мовах, повинна у кожній з цих мов визначати одне і теж, і приводити до однієї і тієї ж поведінки програм;
- позбавлятися характеристик або властивосте, які залежать від платформи або не є універсальними;

- «не платити за те, що не використовується» — ніякий мовний засіб не повинен призводити до зниження продуктивності програм;
- не вимагати занадто ускладненого середовища розробки.

Вибір саме С в якості бази для створення нової мови програмування пояснюється тим, що мова С:

- є багатоцільовою, лаконічною і відносно низькорівневою мовою;
- підходить для вирішення більшості системних завдань;
- виконується будь-де і на всьому;
- стикується з середовищем програмування UNIX [1].

Новими можливостями С++ в порівнянні з С є:

- підтримка об'єктно-орієнтованого програмування через класи. С++ надає всі чотири можливості ООП — абстракцію, інкапсуляцію, успадкування (в тому числі і множинне) і поліморфізм;
- підтримка узагальненого програмування через шаблони функцій і КЛАСІВ;
- стандартна бібліотека С++ складається зі стандартної бібліотеки С (з деякими модифікаціями) і бібліотеки шаблонів (Standard Template Library, STL), яка надає великий набір узагальнених контейнерів і алгоритмів;
- додаткові типи даних;
- обробка винятків;
- віртуальні функції;
- простір імен;
- вбудовувані (inline) функції;
- перевантаження операторів;
- перевантаження функцій;
- посилання та оператори управління вільно розподіляються пам'яттю.

В С++ не дозволяється:

- викликати функцію main() всередині програми, в той час як в С — дія правомірна;

- неявне приведення типів між незв'язаними типами покажчиків;
- використовувати функції, які ще не оголошені.

Переваги мови С++:

- висока сумісність з мовою С;
- обчислювальна продуктивність;
- підтримка різних стилів програмування: структурне, об'єктно-орієнтоване, узагальнене програмування, функціональне програмування, що породжує метапрограмування;
- автоматичний виклик деструкторів об'єктів (в порядку зворотному виклику конструкторів) спрощує і підвищує надійність управління пам'яттю та іншими ресурсами (відкритими файлами, мережевими з'єднаннями, т. п.);
- перевантаження операторів;
- шаблони дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних;
- можливість розширення мови для підтримки парадигм, які не підтримуються компіляторами безпосередньо;
- доступність для мови С++ існує величезна кількість навчальної літератури, перекладеної на всілякі мови.

Недостатки мови С++:

- погано продуманий синтаксис звужує спектр застосовності мови;
- мова не містить багатьох важливих можливостей;
- мова містить небезпечні можливості;
- продуктивність праці програмістів на мові виявляється невиправдано низька;
- громіздкість синтаксису;
- Важка спадщина;
- необхідність стежити за пам'яттю.

5.2.2 Мова Java

Мова Java виникла як частина проекту розроблення передового програмного забезпечення для різних побутових пристроїв. Мовою програмування C++ була розпочата реалізація проекту, але незабаром утворилася група перешкод, ідеальним способом боротьби з якою була заміна самого інструмента — мови програмування. Очевидним стало, що потрібна платформо-незалежна мова програмування, яка дозволяє розробляти програми, які б не доводилося компілювати порізно для кожної архітектури й можна було б застосовувати на різних процесорах під різними операційними системами.

Програми на Java транслюються в байт-код, який виконує віртуальна java-машина (JVM). Вона оброблює байтовий код і віддає інструкції устаткуванню як інтерпретатор, але з тією різницею, що байтовий код, на відміну від тексту, обробляється значно швидше. Достойність такого роду способу виконання програм — в абсолютній незалежності байт-кода від ОС та устаткування, що дає можливість виконувати Java-додатки на будь-якому пристрої, який підтримує віртуальну машину.

Додатки переносяться на безліч різних платформ. Одного разу написаний додаток не доведеться переписувати під інші платформи: він буде працювати без будь-яких змін на різних операційних системах і апаратних архітектурах. Мова Java — об'єктно-орієнтована й одночасно досить проста мова програмування. Процес розроблення програмного забезпечення з використанням Java значно скорочується в силу того, що Java — інтерпретована мова. Довгий процес компіляції-збірки-завантаження застарів, тепер програму тільки треба відкомпілювати та відразу використовувати. Java контролює звернення до пам'яті що додатки робить надійними. Додатки високопродуктивні, незважаючи на те, що мова Java — інтерпретується, код Java програми оптимізується до фази виконання. Підтримка системи багатопоточності дозволяє створювати паралельно виконувальні взаємодіючі легковагі потоки.

Розробнику не потрібно тривалий час вивчати мову, перш ніж він зможе на ній писати код, так як простота мови входить в ключові властивості Java. Фундаментальні концепції мови Java швидко схоплюються і розробники з самого початку можуть вести продуктивну роботу. Розробниками Java було прийнято до уваги, що багато програмістів добре знайомі з мовою C++, тому Java, наскільки це можливо, наближена до C++.

Деякі можливості мови C++ які дуже рідко використовуються, погано розуміються, ускладнюють розробку написання та приносять більше проблем, ніж переваг не включені в Java. Довелося відмовитися від множинного успадкування, перевантаження операторів, але перевантаження методів в Java залишилося.

Автоматична збірка сміття була додана в Java, що спрощує процес програмування, тому що не потрібно слідкувати за очисткою пам'яті, але дещо ускладнює систему в цілому. Завжди викликало масу проблем управління пам'яттю в C і C++, тепер же про це не доведеться багато піклуватися.

Мову Java використовує велика кількість організацій так чи інакше. Часто виникає питання «де використовується Java?». Тому що широкий діапазон варіантів використання Java робить її практично непомітною у використанні. Давайте подивімося, в яких сферах використовується Java:

- програми для Android: незважаючи на дуже активне зростання Kotlin, Java, як і раніше, залишається де-факто головною мовою Android-додатків. Таким чином, всі програмісти Java дуже легко можуть стати розробниками Android-додатків. Хоча Android використовує Android SDK замість JDK, проте, код написаний на Java;
- програмні продукти: Hadoop і Apache Storm, Java використовувалася для створення Eclipse, OpenOffice, Gmail, Atlassian та інших;
- фінансові програми: Java-одна з найбільш затребуваних мов у фінансовій галузі. Вона використовується для створення надійних, швидкісних та простих веб-сайтів як на стороні сервера, так і на стороні клієнта. Java також використовується для моделювання даних;

- касові термінали: багато компаній застосовують Java для створення систем PoS, оскільки їх розробка зазвичай потребує кросплатформенності і великого складу фахівців;
- торгові системи: на Java написана Murex, популярна програма управління банками для фронтального і зворотного зв'язку;
- програми для роботи з великими даними: Hadoop написаний на Java. Scala, Kafka і Spark використовують JVM. Крім того, Java надає доступ до безлічі перевірених бібліотек, інструментів налагодження та моніторингу.

Переваги мови Java:

- сувора статична типізація. Вона значно скорочує чисельність помилок і поліпшує здійсненність підтримки коду, тим паче при застосуванні статичного аналізатора;
- полегшена версія C#. Всі програмісти, які вивчають C++ і Java, дуже подібні до цієї мови. Java вивчається на основі C++ і досить легко. Більш того, Java є елементарним, передбачуваним і мінімалістичним. Нові функції представлені дуже ґрунтовно;
- велика історія. Мова Java існує на ринку з 1995 року, тому сьогодні він добре вивчений і робить відмінну пропозицію сотнями рішень та технологій;
- кросплатформеність. Додатки працюють практично на всьому: починаючи від суперкомп’ютерів до смарт-карт. Особливо добре виконуються на безкоштовному Linux і FreeBSD;
- велика кількість бібліотек. Існує близько 350 гігабайтів бібліотек, доступних в Maven Repository. Можна відшукавши будь для чого завгодно, і, у великій більшості своїй, безкоштовно;
- доступні ціни. Міцне інтегроване середовище розробки, як, наприклад, «IntelliJ IDEA» — вищий щабель еволюції засобів розробки, і вона набагато дешевша, ніж Visual Studio Ultimate для C#;

- прибічники Java. Java — більш в цілому відома мова програмування, що є причиною великого числа Java-програмістів. Для нього також існує офіційна сертифікація SCJP (Sun Certified Java Programmer);
- повна зворотна сумісність. Java має повну зворотну сумісність з API і API ранніх версій. Жодна мова не зможе похвалитися такою сумісністю, навіть C#. Java ефективно застосовується в корпоративній сфері для великих проектів, займаючи дуже міцну позицію.

Недостатки мови Java:

- платне комерційне використання;
- низька продуктивність;
- відсутність нативного дизайну;
- багатослівний і складний код.

5.2.3 Мова C#

Мова програмування C# суміщає об'єктно-орієнтовані і контекстно-орієнтовані концепції програмування. Вона розроблялась в 1998-2001 роках в компанії Microsoft як головна мова розроблення додатків для платформи Microsoft .NET групою інженерів під керівництвом Андерса Хейлсберга. В стандартну установку .NET входить компілятор C#, тому програми на ньому можна створювати й компілювати навіть без інструментальних засобів Microsoft Visual Studio.

Майже 75% синтаксичних можливостей мови C# подібні мові програмування Java, як і Java, C# спочатку призначався для веб-розробки. Його крім того також називають «очищеною версією Java». Біля 10% синтаксичних можливостей запозичено з мови C++ та 5% — з Visual Basic. І близько 10% — це реалізація власних ідей розробників C#.

Всупереч на дуже суттєві розбіжності між компонентною об'єктною моделлю COM (головного стандарту Microsoft для компонентного проектування і реалізації

програмного забезпечення) і моделлю Java, мова програмування має досить багато спільного.

Загальне середовище виконання програм засноване на використанні проміжної мови IL (Intermediate Language — проміжна мова), яка здійснює майже ту ж роль, що і байт-код віртуальної машини Java. Застосовані в рамках технології .NET компілятори з різних мов транслюють програми в IL-код. Так само, як і байт-код Java, IL-код являє собою команди гіпотетичної стекової обчислювальної машини. Але є і різниця в пристрої й застосування IL.

По-перше, на відміну від JVM, IL не прив'язаний до однієї мови програмування. В складі минулих версій Microsoft.NET є компілятори мов C++, C#, Visual Basic. Незалежні розробники також можуть додавати інші мови, створюючи свої компілятори в IL-код.

По-друге, IL розрахований не для програмної інтерпретації, а для подальшої компіляції в машинний код. Це дає можливість досягти істотно більшої швидкодії програм. Файли які містять IL-код несуть достатньо інформації для роботи оптимізующого компілятора.

Кік Редек з Microsoft вважає що мова C# більш складною мовою, ніж Java. На його думку, «мова Java була побудований таким чином, щоб уберегти розробника від пострілу в ногу», а «C# був побудований так, щоб дати розробнику пістолет, але залишити його на запобіжнику».

В основу проектування лягла легкість застосування при розробці мови, домінуючи над потужністю мови й швидкістю виконання. З цього місця і збирач сміття з контрольованими об’єктними посиланнями, який автоматично звільняє пам’ять, відбираючи при цьому процесорний час. На думку багатьох розробників, другим найважливішим чинником уникнення помилок при розробці є безпека роботи з типами.

Переваги мови C#:

– лямбда-функції. Як і анонімні внутрішні класи Java, C# гарантує більш продуктивні лямбда-функції. Це більш практичний метод для установлення об’єкта анонімної функції безпосередньо в точці виклику

або надання функції як аргумент. Як правило, лямбда-вирази застосовуються для інкапсуляції передаються в алгоритми або асинхронних методів декількох рядочків коду;

- оператор перевантаження. Мова C# дає змогу застосовувати перевантаження операторів для їх використання у власних класах. Це дає змогу досягти природного вигляду визначеного користувачем типу даних і застосовувати як основний тип даних;
- делегати застосовуються в якості методів для вказівки на інші методи. В Java теж є така функція, але в C# це простіше;
- властивості C# надають кожному класу можливість забезпечити загальний засіб отримання й установки значень, приховуючи при цьому код реалізації або верифікації. Це дає можливість даним бути легкодоступними і продовжувати просувати безпеку і гнучкість методів;
- підтримка ключового слова «yield». За допомогою «yield» можна встановити генератор. Використовуючи генератор, можна створювати обчислення послідовностей «на льоту», що допомагає заощадити пам’ять і виконувати обчислення нескінченних послідовностей;
- розширення методів — досить зручний спосіб розширення існуючих класів, навіть якщо вони готові без реального розширення;
- підтримка оператора «?» пропонує досить простий синтаксис для отримання значення посилального типу.

Недостатки мови C#:

- пріоритетна орієнтованість на Windows платформу;
- мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів та учнів. Великій компанії покупка ліцензійної версії цієї мови обійдеться в круглу суму;
- у мові залишилася можливість використання оператору безумовного переходу.

5.2.4 Вибір мови програмування

Сьогодні в більшості вузів нашої планети викладають мову Java в якості першої мови програмування з педагогічних причин. Достойність Java в тому, що вона послідовна, дуже добре структурована й переносима. Але це не швидко, тому що в більшості випадків це працює на віртуальній машині Java (JVM), яка інтерпретує код, а не виконує його. Якщо важлива швидкість виконання, то кращим вибором буде C++. Мова C++ має більше можливостей та варіантів, ніж більшість інших мов програмування. Розширені функції, такі як успадкування, поліморфізм, макроси, бібліотеки шаблонів й обробка винятків, дозволяють створювати добре структурований та повторно використовуваний код на високому рівні абстракції.

5.3 Технології розпаралелювання для мови C++

Сучасні технології для паралельного програмування відрізняються одна від одної не стільки мовами програмування, скільки архітектурними підходами до побудови паралельних систем. Наприклад: деякі технології пропонують побудову паралельних програм на основі декількох комп'ютерів; а інші ж дозволяють працювати на одній машині з кількома процесорами.

Для паралельного програмування на основі C++ використовуються такі бібліотеки, як OpenMP, MPICH (реалізація бібліотеки MPI.

5.3.1 Технологія OpenMP

Технологія OpenMP (Open Multi-Processing) це найбільш популярна і застосовуваний сьогодні розробка розпаралелювання. За базу береться послідовна програма, а для її розпаралелювання застосовується набір директив компілятора, змінні оточення й бібліотечні процедури, які призначені для програмування багатопоточних додатків на багатопроцесорних системах з загальною пам'яттю.

Стандарт OpenMP був запроваджений на мові Fortran в 1997 р., але пізніше в 1998 р. вмістив в себе і мови C/C++. ARB (Architecture Review Board) — некомерційна організація яка займається розробкою OpenMP, до її структури входять найбільші розробники SMP-архітектур та програмного забезпечення.

OpenMP — комплект спеціальних директив компілятора, змінних оточення та бібліотечних функцій. Найбільш оригінальні директиви компілятора, які застосовуються для відмітки ділянки в коді з можливістю паралельного виконування. Компілятор, який підтримує OpenMP, перетворює вихідний код та вставляє відповідні виклики функцій для паралельного виконання цих ділянок коду.

Для розробників які хочуть швидко розпаралелити свої обчислювальні програми з великими паралельними циклами, найкраще підходить OpenMP, за рахунок ідеї «часткового розпаралелювання». Розробнику непотрібно створювати нову паралельну програму, а просто потрібно додати в текст послідовної програми OpenMP директиви. Передбачено, що OpenMP-програма на однопроцесорній платформі може бути застосована в якості послідовної програми, тому що немає необхідності одночасно підтримувати послідовну та паралельну версії. Тому що послідовний компілятор просто ігнорує OpenMP директиви, а для виклику OpenMP процедур можуть бути підставлені заглушки.

OpenMP простий у застосуванні та включає лише дві основні типи конструкцій: директиви pragma й функції виконуючого середовища OpenMP. Як реалізувати паралельне виконання блоків коду, як правило, вказують компілятору директиви pragma. З фрази pragma omp починаються всі директиви. Декілька додаткових атрибутів може мати кожна директива. Роздільно специфікуються атрибути для призначення класів змінних, які можуть бути атрибутами різних директив.

Програма, розроблена із використанням технології OpenMP, складається з послідовних (однопоточних) і паралельних (багатопоточних) ділянок. В OpenMP використовується модель розпаралелювання «Розгалуження – Злиття». Спочатку є тільки єдиний потік його називають початковим потоком або ще основною ниткою (Master thread, тут threard це легковагові потоки). Як тільки в коді програми потік

зустрічає паралельну конструкцію, він створює групу потоків і стає головним потоком. У створеній групі всі потоки, включаючи головний, виконують код програми. Після виконання паралельної конструкції в коді, роботу продовжує тільки головний потік.

Кількість потоків які виконують паралельну ділянку можливо контролювати по різному, можна використовувати OMP_NUM_THREADS або викликати процедуру omp_set_num_threads().

OpenMP припускає, що програма виконується в системі з роздільною пам'яттю, в якій зберігаються змінні програми доступні всім її ниткам. Крім того, кожна нитка має доступ до пам'яті, недоступною для всіх інших ниток; така пам'ять називається приватною пам'яттю нитки.

Пам'ять в OpenMP розділяється на: локальну та загальну пам'ять. Локальна пам'ять доступна тільки для однієї нитки, а загальна, для кількох.

Основні можливості й позитивні якості технології розпаралелювання OpenMP:

- надається можливість реалізувати максимально високоефективно багатопроцесорні обчислювальні системи із загальною пам'яттю, дозволяючи застосовувати загальні дані для паралельно виконувальних потоків, без будь-яких труднощів для міжпроцесорних передач даних;
- надається можливість поетапного розроблення паралельних програм. Це означає, що на ранніх стадіях розробки можна отримати паралельну програму, це досягається завдяки директивам OpenMP, вони можуть додаватися у послідовну програму покроково;
- знижена ймовірність виникнення появи перешкод при перенесенні паралельних програм між різними комп'ютерами. Програма, написана на мові C або Fortran з застосуванням технології розпаралелення OpenMP, буде виконуватися для різних обчислювальних систем із загальною пам'яттю;
- досить простий перелік директив для вивчення. Розроблення порівняно простих паралельних програм не вимагає великих зусиль, іноді достатньо

додати декілька директив в послідовну програму. Але не варто забувати, що при розробленні складних програм потрібно мати відповідні знання.

Недоліки OpenMP

- обмеженість його області застосування (мультипроцесори й DSM-кластери);
- наявні в ньому засоби розпаралелювання циклів із залежностями за даними, є занадто низькорівневими;
- розробнику по суті дозволяється використовувати безпосередньо модель виконання (програмувати в термінах ниток), що може провокувати створення погано переносимих програм.

5.3.2 Технологія MPI

MPI розшифровується як «Message passing interface» (Інтерфейс передачі повідомлень). MPI — це стандарт на програмний інструментарій для забезпечення зв'язку між окремими процесами паралельної програми. MPI надає розробнику єдиний інструмент взаємодії процесів всередині паралельного завдання, що виконується незалежно від апаратної архітектури (однопроцесорні, багатопроцесорні зі спільної чи роздільною пам'яттю), взаємного розташування процесів (на одному фізичному процесорі або на різних) і API операційної системи. Програма, що застосовує MPI, легко налагоджується і переноситься на інші платформи.

Рішення про застосування у своїх проектах MPI слід приймати обережно, після ретельного зважування своїх сил й можливостей як програміста. Не зважаючи на те, що MPI являє собою значний хід вперед в порівнянні з попереднім поколінням бібліотек передачі повідомлень, а можливо і внаслідок цього, програмувати на MPI досить складно. Причиною тому являється не недолік стандарту, а в самій ідеології передачі повідомлень. MPI можна розглядати як рівень асемблера для паралельних програм.

В даний час різними командами розробників написано декілька програмних пакетів, що задовольняють специфікації MPI, зокрема: MPICH, LAM, HPVM,

OpenMPI і так далі. У декілька словах охарактеризуємо найбільш поширені з цих пакетів. Якщо говорити про LAM, то основна перевага цього пакету-багаті налагоджувальні можливості. Трасування звернень до MPI та дослідження становища паралельної програми після аварійного завершення, роблять процес налагодження менш важким і більш продуктивним. З іншого боку, пакет MPICH більш мобільний, придержуючись простим інструкціям можна перенести MPI на нову платформу (наприклад з Linux на Windows або навпаки). Для цього необхідно лише кілька драйверів нижнього рівня написати. Установка бібліотеки MPICH проходить трохи складніше, ніж установка LAM MPI, оскільки доводиться задавати набагато більшу кількість параметрів, причому деякі з них відомі лише розробникам.

Open MPI [4] являє собою об'єднання трьох відомих реалізацій MPI:

- FT – MPI з Університету Теннессі;
- LA – MPI з Лос-Аламоської Національної Лабораторії;
- LAM / MPI з Університету Індіана;

Розробники OpenMPI обрали ці реалізації MPI, так як вони виділяються в одній або декількох областях. OpenMPI бажає використовувати кращі ідеї та технології окремих проектів та створити одну реалізацію MPI з відкритим вихідним кодом світового класу, який виділявся у всіх областях. Проект OpenMPI визначає кілька найважливіших цілей (цілей верхнього рівня):

- створення вільної рецензованої повної реалізації MPI з відкритим вихідним кодом;
- забезпечення надзвичайно високої, конкурентоспроможної продуктивності (малий час затримки або висока пропускна здатність);
- залучення високопродуктивної обчислювальної спільноти безпосередньо з зовнішньої розробкою і зворотним зв'язком;
- забезпечення сталої платформи для 3rd-party дослідження і комерційного розвитку;

- допомога в запобіганні «проблеми породження», притаманний для інших проектів MPI;
- підтримування широкого спектру високопродуктивних обчислювальних платформ та середовищ [5].

Переваги MPI:

- спроможність використання в мовах Фортран, C, C++;
- надання можливостей для поєднання обмінів повідомленнями та обчислень;
- надання режимів передачі повідомлень, що дозволяють уникнути зайвого копіювання інформації для буферизації;
- широкий набір колективних операцій (наприклад, збір інформації з різних процесорів, що допускають ефективну реалізацію;
- широкий набір редукційних операцій (наприклад, підсумовування розташованих на різних процесорах даних, або знаходження їх максимальних або мінімальних значень), що спрощують роботу програміста і допускають ефективну реалізацію;
- зручні засоби іменування адресатів повідомлень, що спрощують розробку стандартних програм або поділ програми на функціональні блоки;
- можливість завдання типу переданої інформації, що дозволяє забезпечити її автоматичне перетворення в випадку розходжень в поданні даних на різних вузлах системи.

Недоліки MPI:

- складний в освоєнні і використанні для досягнення ефективної взаємодії між ними за рахунок недостатньо якісного коду й де-факто, часткової підтримки стандарту;
- портування однієї з існуючих реалізацій під нову архітектуру може бути складним через низькорівневого стандарту;
- для запуску програми необхідна установка відповідного програмного забезпечення на всі складові програмного комплексу;

– вимагає більше програмування для переходу з послідовної в паралельну версію.

5.3.3 Вибір технології розпаралелювання

В даному підрозділі наведено короткий огляд комплексів API, призначених для реалізації розподілених обчислень в різних середовищах і архітектурах. Зрозуміло, у кожному конкретному випадку вибір відповідного API повинен робитися виходячи з наявного обладнання, проте в загальному можна рекомендувати наступне:

– при необхідності вирішення завдань розподілених обчислень на базі SMP-систем, в якості бази доцільно вибирати OpenMP;
– MPI слід використовувати тільки в разі наявності в команді фахівця, що має досвід роботи з цим API — бувши архітектурно досить складним комплексом, в невмілих руках він здатний дати менше прискорення.

5.4 Процесор Intel Xeon Phi 7210

Intel Xeon Phi — x86 процесор розроблений компанією Intel. Даний процесор призначений для застосування на суперкомп'ютерах, серверах та високопродуктивних робочих станцій. Він як ніхто інший, відкриває всі переваги паралельного виконання програм. Розроблений за технологією Intel Many Integrated Core (MIC), він надає декілька десятків потужних обчислювальних ядер і достойний кусок інтегрованої високошвидкісної пам'яті. Його архітектура дозволяє використовувати стандартні мови програмування й існуючі програмні засоби розпаралелювання, такі як OpenMP.

Даний процесор забезпечує високопродуктивне обчислення, яке потрібне щоб забезпечити безпрецедентну продуктивність для інноваційних проривів у виробництві, науках про життя, енергетиці та інших галузях. Здатність швидко обчислювати, моделювати й приймати більш обґрунтовані рішення стимулювала зростання високопродуктивних обчислень і аналітики. Це було викликано

глобальними пріоритетами бізнесу і досліджень для більш точного прогнозування погодних умов, створення більш ефективних енергетичних ресурсів і розробки методів лікування хвороб серед багатьох інших нагальних проблем. Завдяки проривній продуктивності та іншим новим характеристикам співпроцесора Intel Xeon Phi промисловість отримає ще більшу надійність в отриманні точних відповідей, допоможе поширити високопродуктивні обчислення за межі лабораторій і університетів і досягти максимальної продуктивності [8].

Всі ці області зараз використовують апаратні засоби AMD і Nvidia для нарощування обчислювальної потужності. Intel просто робить те ж саме, тільки її продукція не вимагає перероблювання й написання коду для CUDA або OpenCL [7].

Intel розпочала продажі Xeon Phi 7210 в червні 2016.

Технічні характеристики процесора Intel Xeon Phi 7210 [6]:

- тип — серверний;
- кодова назва архітектури — Knights Landing;
- кількість ядер — 64;
- базова тактова частота — 1.30 ГГц;
- максимальна тактова частота — 1.50 ГГц;
- кеш 1-го рівня — 32 Кб (на ядро);
- кеш 2-го рівня — 512 Кб (на ядро);
- технологічний процес — 14 нм;
- кількість транзисторів — 8000 млн;
- Підтримка 64 біт — так;
- допустима напруга ядра — 0.550-1.125 V;
- типи оперативної пам'яті — DDR4-2133;
- допустимий обсяг пам'яті — 384 Гб;
- кількість каналів пам'яті — шість;
- пропускна здатність пам'яті — 102 Гб / с.

6 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

В поданому розділі аналізується використання апарату САА-М та системи ІПС для проектування паралельної програми розв’язання задачі тривимірної конвективної дифузії, яка з'являється при математичному моделюванні в метеорології атмосферної циркуляції.

6.1 Математична модель

Детально описана в [21] постановка задачі конвективної дифузії. Модель циркуляції атмосфери розглянута в згаданій роботі, яка описується такими головними фізичними законами:

а) збереження чисельності руху

$$\frac{dV}{dt} = -\frac{1}{\rho} grad p - 2\Omega \times V + F_g + \frac{1}{\rho} F_f, \tag{6.1}$$

б) збереження маси (нестисливість середовища)

$$div V = 0, \tag{6.2}$$

в) збереження теплової енергії

$$\frac{dT}{dt} = \frac{\delta}{c_v \rho}, \tag{6.3}$$

г) рівняння стану повітря

$$p = \rho R_c T, \tag{6.4}$$

де V – вектор швидкості у неінерційній системі координат;

p — атмосферний тиск;

ρ — густина повітря;

F_g — сила тяжіння;

Ω — вектор кутової швидкості обертання Землі;

δ — притік тепла до одиниці об’єму повітря;

F_f — щільність сили тертя;

T — абсолютна температура;

R_c — питома газова стала сухого повітря.

c_v — питома теплоємність сухого повітря за сталого об’єму;

Які характеризують макромасштабну циркуляцію в моделях, застосовують сферичну систему координат. Рівняння (6.1) – (6.3) після деяких спрощень матимуть вигляд

$$\frac{\partial u}{\partial t}+\frac{u}{a\cos\varphi}\frac{\partial u}{\partial \lambda}+\frac{v}{a}\frac{\partial u}{\partial \varphi}+w\frac{\partial u}{\partial z}=-\frac{1}{\rho a\cos\varphi}\frac{\partial p}{\partial \lambda}+(v\sin\varphi-w\cos\varphi)\times \times\left(2\omega+\frac{u}{a\cos\varphi}\right)+\frac{1}{\rho}F_{\lambda}, \tag{6.5}$$

$$\frac{\partial v}{\partial t}+\frac{u}{a\cos\varphi}\frac{\partial v}{\partial \lambda}+\frac{v}{a}\frac{\partial v}{\partial \varphi}+w\frac{\partial v}{\partial z}=-\frac{1}{\rho a}\frac{\partial p}{\partial \varphi}-u\left(2\omega+\frac{u}{a\cos\varphi}\right)\sin\varphi-\frac{wv}{a}+\frac{1}{\rho}F_{\varphi}, \tag{6.6}$$

$$\frac{1}{a\cos\varphi}\frac{\partial u}{\partial \lambda}+\frac{1}{a}\frac{\partial v}{\partial \varphi}+\frac{\partial w}{\partial z}-\frac{v}{a}tg\varphi=0, \tag{6.7}$$

$$\frac{\partial T}{\partial t}+\frac{u}{a\cos\varphi}\frac{\partial T}{\partial \lambda}+\frac{v}{a}\frac{\partial T}{\partial \varphi}+w\frac{\partial T}{\partial z}=\frac{\delta}{c_V\rho}, \tag{6.8}$$

де λ — довгота;

z — висота над рівнем моря;

φ — широта;

a — радіус Землі;

$V = (u, v, w)$;

ω — швидкість добового обертання Землі;

$F_f = (F_\lambda , F_\varphi , F_z)$.

Відповідно до трирівневого алгоритму [22] відбувається паралельне чисельне здійснення даної моделі, за рівняннями, підобластями та просторовими напрямками, передбачається розпаралелювання. Застосовується модифікований адитивно-усереднений метод розщеплення (МАУМ) [13] алгоритму. До розрахункової області використовуються покоординатні декомпозиції: за напрямками φ та z для підзадач розподілення уздовж λ; для підзадач за напрямком λ розподілення уздовж φ. Саме тому в програмі застосовані два формати представлення даних в масивах: основний [λ][φ][z] та допоміжний [φ][z][λ]. Із рівнянь (6.5) та (6.6) складалися горизонтальні складові руху та, із (6.7) – вертикальна складова *w*, із (6.8) – температура T, а із (6.4) — густина ρ.

6.2 Схема паралельного алгоритму

В поданому підрозділі представлена спроектована САА-схема паралельного алгоритму вирішення зазначеної вище задачі конвективної дифузії. Схема розроблена з застосуванням інструментальної системи ІПС [12-16]. На базі схеми зроблена генерація в програмний код на мови програмування С++ з застосування інструментів OpenMP [24]. З двох складених операторів "Основна схема" й "Паралельні обчислення" сформована САА-схема, яка представлена на рис. 6.1 та рис. 6.2. Ці складові оператори відповідають функціям *main* і *CalcParallelPart* в програмі на мові С++.

В складеному операторі "Основна схема" здійснюється виклик операторів ініціалізації даних (змінних й масивів), підрахунок часу виконування програми, виконання паралельних обчислень складеного оператора, зіставлення розрахункових значень та збереження отриманих результативних даних у файли.

У цій схемі вказані такі змінні:

– *Proc_Num* — чисельність паралельних потоків;

– *Subdomains* — чисельність підобластей;
– *TmLimCalc* — в умові m-циклу кінцеве значення часу;
– *M_prm* — *m*-параметр МАУМ;
– *TmKnotsPer12h* — для 12-годинного періоду чисельність часових точок ;
– *tau* — часовий крок;
– *U*, *V* — горизонтальні складові руху;
– *W* — вертикальна складова;
– *P* — атмосферний тиск;
– *T* — абсолютна температура.

Схема "Паралельні обчислення" (див. Рис. 6.2) початок відбувається з встановлення початкових значень змінних й масивів, потім відбувається виконання m-циклу МАУМ. Спочатку відбувається готування головних розрахунків в тілі m-циклу: установлення граничних умов, розрахунок коефіцієнтів рівнянь, вертикальної складової швидкості. Потім для кожного з напрямків λ, φ та z обчислюються підзадачі. Далі зі збереженням граничних умов які відповідають різним напрямкам, проводиться усереднення результатів. Отримані підсумки в кінці циклу записуються в загальні масиви.

```
СХЕМА ПАРАЛЕЛЬНИЙ АЛГОРИТМ ДЛЯ РОЗВЯЗАННЯ 3-ВИМІРНОЇ ЗАДАЧІ  КОНВЕКТИВНОЇ
        ДИФУЗІЇ
"Основна схема"
==== "Визначити змінну (Proc_Num) типу (int) = (1)"
        ПОТІМ
        "Встановити початкові параметри (Subdomains, M_prm, TmKnotsPer12h,
                                                TmLimCalc, tau)"
        ПОТІМ
        "Почати відлік часу"
        ПОТІМ
        "Завантажити дані в масиви для (P, W, U, V, T)"
        ПОТІМ
        (Proc_Num := "Паралельні обчислення")
        ПОТІМ
        "Закінчити відлік часу та вивести результат"
        ПОТІМ
        "Порівняти інтерпольовані та обчислені значення в нормі L2"
        ПОТІМ
        "Зберегти дані у файли для 3-вимірної задачі конвективної дифузії"
```

Рисунок 6.1 — Початок САА-схеми для задачі конвективної дифузії: складений оператор "Основна схема"

```
"Паралельні обчислення"
==== "Визначити константу (sdm_sz) типу (int) = ((SZ_X1-1)/Subdomains+1)"
        ПОТІМ
        "Визначити змінну (NmbThreads) типу (int)"
        ПОТІМ
        "Визначити масив (pOut[DIR][EVL_MT_VAL][MAX_SUBDMN]) типу (double)"
        ПОТІМ
        "(NmbThreads) := (Subdomains * DIR * EVL_MT_VAL)"
        ПОТІМ
        ПАРАЛЕЛЬНО (proc = 0, ..., NmbThreads-1)
        (
                "Ініціалізація змінних та масивів"
                ПОТІМ
                "Коментарій (***** m-цикл *****)"
                ПОТІМ
                "(n) := (M_prm)"
                ПОТІМ
                ПОКИ '(n * tau) <= (TmLimCalc)'
                ЦИКЛ
                        "Встановити значення для U, V, T, P, D, W, F, C та граничні умови"
                        ПОТІМ
                        ВИБІР ['(dir) = (0)']
                                "Обчислити задачі для напрямку (Lam)",
                                ['(dir) = (1)']
                                "Обчислити задачі для напрямку (Fi)",
                                ['(dir) = (2)']
                                "Обчислити задачі для напрямку (Z)")
                        ПОТІМ
                        "Поміняти місцями значення в масивах pR1 та pR2"
                        ПОТІМ
                        ЧЕКАТИ 'Обробка в усіх потоках закінчена'
                        ПОТІМ
                        "Обчислити середні значення на основі отриманих результатів для
                        кожного напрямку"
                        ПОТІМ
                        ЧЕКАТИ 'Обробка в усіх потоках закінчена'
                        ПОТІМ
                        "Занести результати в глобальні масиви"
                        ПОТІМ
                        ЧЕКАТИ 'Обробка в усіх потоках закінчена'
                        ПОТІМ
                        "Збільшити (n) на (M_prm)"
                КІНЕЦЬ ЦИКЛУ
                ПОТІМ
                "Звільнити пам'ять для масивів для 3-вимірної задачі конвективної дифузії"
        )
        ПОТІМ
        "Повернути значення (NmbThreads)";
```

Рисунок 6.2 — Продовження САА-схеми для задачі конвективної дифузії:

деталізація складеного оператора "Паралельні обчислення"

Окрім вищезазначених, у схемі паралельних обчислень вказані такі додаткові змінні та константи:

- *proc* — номер потоку;
- *NmbThreads* — чисельність паралельних потоків;
- *sdm_sz* — в напрямку λ розмір підобластей;
- *SZ_X1* — у напрямку λ чисельність точок скінченно-різницевої сітки;
- *pOut, pR1, pR2* — для збереження проміжних результатів;
- *DIR* = 3 кількість напрямків (λ, φ, *z*);
- *EVL_MT_VAL* = 3 — в еволюційному рівнянні чисельність метеорологічних величин ;
- *MAX_SUBDMN* — максимальна чисельність підобластей;
- *C* — адвекція;
- *n* — змінна *m*-циклу;
- *F* — коефіцієнт рівнянь;
- *D* — густина повітря;
- *dir* — для збереження значення поточного напрямку (дорівнює 0 для λ; 1 для φ; 2 для *z*).

В алгоритмі чисельність паралельних потоків встановлюється згідно формули

$$(NmbThreads) = (Subdomains * DIR * EVL\_MT\_VAL). \quad (6.9)$$

На фундаменті побудованої САА-схеми в системі ІПС відбулося здійснення генерації програмного коду на C++ із застосуванням технології OpenMP [24]. ДСП-конструктор інструментарію ІПС у процедурі генерації коду застосовує шаблони програмних реалізацій операцій САА-М та базисних понять із бази даних. Для даної задачі (ПАРАЛЕЛЬНО) була реалізована операція асинхронного виконання потоків за допомогою застосування директиви OpenMP #pragma omp parallel. Була

використана OpenMP директива #pragma omp barrier для здійснення процедури синхронізації (ЧЕКАТИ), яка здійснює затримку обчислень до завершення роботи всіх потоків.

В розділі 7 розглядаються результати проведеного експерименту з виконанням реалізації згенерованої програми на багатопроцесорній платформі.

6.3 Програмна реалізація

Для реалізації цієї програми використовувався процедурний стиль написання програм. Алгоритм роботи програми представлений у вигляді діаграм та блок схем в додатках В, Г, Д, Е, Ж.

Перелік головних функцій та процедур:

- CalcParallelPart — обчислення задачі 3-вимірної конвективної дифузії;
- LoadDataToArrays — завантаження даних в масиви;
- SaveDataToFiles — збереження даних у файли;
- InitLocalArrays — ініціалізація локальних масивів;
- SetActualUV — встановити значення для горизонтальних складових руху;
- SetActualT — встановити значення для абсолютної температури;
- SetActualW — вертикальна складова;
- RunCalcFi — обчислити завдання для напрямку (Fi);
- RunCalcLam — обчислити завдання для напрямку (Lam);
- RunCalcZ — обчислити завдання для напрямку (Z);
- CompareRes — порівняти інтерпольовані та обчислені значення в нормі L2.

7 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ

Здійснено експеримент [11] з застосуванням спроектованої паралельної програми для вирішення тривимірної задачі конвективної дифузії на 64-ядерному процесорі Intel Xeon Phi CPU 7210, 1.30 ГГц.

В ході експерименту були установлені такі значення для параметрів програми:

- *TmLimCalc* = 12 годин — термін, на який здійснюється розрахунок прогнозу погоди;
- *M_prm* = 10 — параметр МАУМ;
- *tau* = 10 секунд — часовий крок;
- *Subdomains* — чисельність підобластей змінювалась від 1 до 7, відповідно кількість паралельних потоків *NmbThreads* = *Subdomains* * 9 встановлювалась від 9 до 63.

Залежність мультипроцесорного прискорення визначається як

$$Sp = \frac{T_1}{T_N}, \tag{7.1}$$

де T_1 — час виконання послідовної програми;

T_N — час виконання паралельної програми на N ядрах процесора, N = NmbThreads.

Максимальне значення прискорення Sp = 58.5 отримане при кількості потоків 63, ефективність використання ядер визначається як

$$E = \frac{Sp}{N} = 0{,}93, \tag{7.2}$$

де Sp — залежність мультипроцесорного прискорення;

N — кількість паралельних потоків.

На рисунку 7.1 представлений графік залежністі мультипроцесорного прискорення від кількості паралельних потоків для паралельної програми розв’язання тривимірної задачі конвективної дифузії.

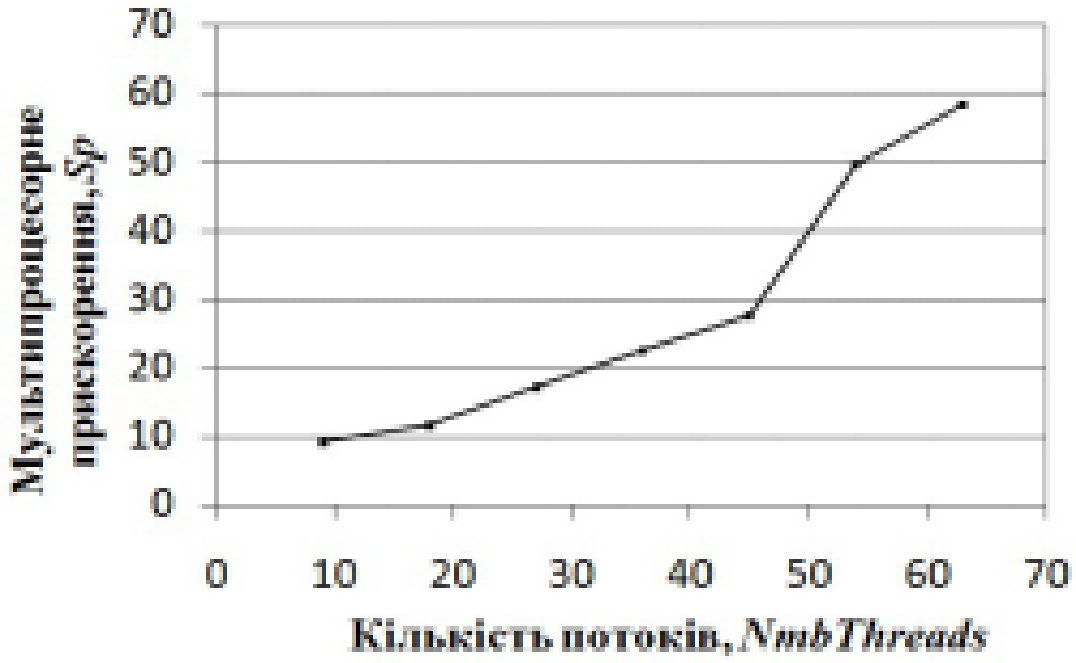

Рисунок 7.1 — Залежність мультипроцесорного прискорення від кількості паралельних потоків для паралельної програми розв’язання тривимірної задачі конвективної дифузії

8 СТАРТАП-ПРОЕКТ

8.1 Опис ідеї проекту

Даний розділ включає економічне обґрунтування стартап-проекту «Засоби розроблення паралельних програм для метеорологічного прогнозування». Розділ спрямований на ознайомлення з економічними та функціональними характеристиками майбутнього проекту, економічними підходами його реалізації та впровадження у застосування.

Опис основної ідеї мого майбутнього стартап-проекту наведено в Таблиці 8.1.

Таблиця 8.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Чисельна модель прогнозування погоди — комп’ютерна пара-лельна програма, яка працює на суперкомп’ютері й забезпечує точні прогнози погоди на багато атмосферних змінних, таких як, температура, тиск, вітер і кількість опадів	Будівельна галузь	Ттаким чином, при отриманні інформації про погіршення погодних умов будівельники припиняє роботу на висоті
	Судноплавство	Завдяки своєчасним штормовим попередженням врятовано сотні морських та річкових суден, тисячі людських життів
	Авіація	Точний прогноз погоди дозволяє розробити перелік профілактичних заходів щодо запобігання аварій
	Сільське господарство	Початок посіву, перенесення посіву на кілька днів через прогнозовані заморозки запобігає втратам мільйонів гривень

Для оцінки конкурентоспроможності та можливості й труднощів виходу стартапу на ринок було проведено порівняння з низкою потенційних конкурентів,

для порівняння з якими можна вибрати характеристики. Результати порівняння наведені в таблиці 8.2.

Таблиця 8.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	К-т 1	К-т 2	К-т 3			
1	Собівартість	Низька	Висока	Середня	Низька			+
2	Наявність інтернету	Треба	Треба	Треба	Треба		+	
3	Зручність використання	Зручно	Зручно	Зручно	Зручно		+	
4	Швидкість роботи	Середня	Швидка	Середня	Середня		+	
5	Масштабованість	Так	Ні	Ні	Ні			+
6	Кросплатформність	Так	Ні	Ні	Ні			+

Проект має сильні сторони, які відсутні в існуючих аналогах і здатний конкурувати з ними. Сильні сторони — низька вартість впровадження і кросплатформеність, що дозволяє швидко почати роботу. Таку перевагу як масштабованість відсутній у потенційних конкурентів, тому є головною перевагою проекту і дозволяє без зайвих зусиль розширити його новими сервісами.

8.2 Технологічний аудит ідеї проекту

Проводиться аудит технологій, за допомогою яких може бути реалізована ідея проекту (Технологія створення товарів). Ухвалення технологічної можливості задумки проекту передбачає аналіз таких складових: яка технологія буде використана для виготовлення товарів відповідно до ідеї проекту, чи існують такі технології та доступні такі технології авторам проекту.

Таблиця 8.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення чисельної моделі прогнозування погоди	C++, OpenMp	Наявна	Безкоштовна, доступна
		Java, fork/join	Наявна	Безкоштовна, доступна
		C#, TPL	Наявна	Частково платна, доступна
Обрана технологія реалізації ідеї проекту: C++, OpenMp які є безкоштовними, доступними та добре дослідженими потенційними розробниками.				

У таблиці 8.3 наведено огляд основних технологічних стеків, які можуть бути використані для реалізації описаної вище системи стартап-проекту. Був обраний стек технологій, який не вимагає додаткових доопрацювань і витрат.

8.3 Аналіз ринкових можливостей запуску стартап-проекту

Проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку. У таблиці 8.4 представлені результати аналізу характеристик попиту на ринку. На ринку три основних представники, і загальний обсяг продажів значний. Однак зростання динаміки ринку і відсутність істотних обмежень для виходу на ринок дозволяють виконати стартап-проект з характеристиками, наведеними в попередніх підрозділах.

Таблиця 8.4 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	13500 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає

Продовження таблиці 8.4

5	Специфічні вимоги до стандартизації та сертифікації	Відсутні.
6	Середня норма рентабельності в галузі (або по ринку), %	15%

Дії, необхідні для виходу на такий ринок, залежать, серед іншого, від потенційних споживачів. Аналіз цільових аудиторій споживачів даної продукції наведено в таблиці 8.5.

Таблиця 8.5 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба отримання точного прогнозу погоди для продовження робіт на висоті	Будівельна галузь	Стартап переважно буде виконувати функції отримання точного корткострочного або довгострочного прогнозу погоди	Зручність у використанні. Швидка робота системи. Спроможність швидко освоїти як користуватися системою.
2	Потреба отримання точного прогнозу погоди для збереження сотні людських життів та суден	Судноплавство	Стартап переважно буде виконувати функції отримання точного корткострочного або довгострочного прогнозу погоди	
3	Потреба отримання точного прогнозу погоди для вчасної розробки попереджувальних дій для запобігання аварійних ситуацій	Авіація	Стартап переважно буде виконувати функції отримання точного корткострочного або довгострочного прогнозу погоди	

Продовження таблиці 8.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
4	Потреба отримання точного прогнозу погоди для початку посівних робіт або перенесення	Сільське господарство	Стартап переважно буде виконувати функції отримання точного корткострочного або довгострочного прогнозу погоди	

За результатами аналізу цільових аудиторій споживачів продукту, описаного в таблиці 8.5 і необхідного для виходу на такий ринок, який був описаний в таблиці 8.4, слід направити зусилля на активне просування проекту.

Важливим процесом виходу на ринок і орієнтації на конкретну цільову аудиторію споживачів є аналіз можливих загроз для стартап-проекту, які можуть викликати суттєві проблеми для його розвитку. Результати відповідного аналізу факторів загрози продукції наведено в таблиці 8.6.

Таблиця 8.6 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на майбутнє. Ітеративна розробка продукту для покрокового запуску продукту на ринок і отримання відповіді користувачів
2	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Зменшення ціни на поставлену послугу. Розробка унікальних характеристик товару; Надання ліцензій на обслуговування
3	Динаміка ринку	Уповільнення росту ринку	Співпраця з іншими компаніями для покращення

			ситуації на ринку

Продовження таблиці 8.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
4	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій. Проведення рекламної компанії
5	Держава	Зростання податкового тягаря	Перегляд виконання умов, що зменшують податки. Поступове підвищення тарифів

За результатами аналізу можливих факторів загроз стартап-проекту описуваного продукту, наведених у таблиці 8.6 і необхідних для виходу на ринок, описаних в таблиці 8.4, існує ряд ризиків, які слід враховувати при плануванні виходу продукту на ринок і мати орієнтовні сценарії по їх мінімізації й компенсації їх впливу, наведені в таблиці вище.

Подібно загрозам стартап-проекту, які можуть викликати значні проблеми для його розвитку, важливою частиною є огляд можливих сприятливих умов, використання яких може значно поліпшити стан спартап-проекту і забезпечити перевагу перед конкурентами. Такі сприятливі умови й можливості описані в таблиці 8.7.

Таблиця 8.7 — Зміст можливості

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зворотній зв’язок від користувачів	Можливість одержання потрібної інформації для поліпшення продукту	Наявність вхідних даних і відповідей на них від команди розробників для задоволення потреб кінцевих користувачів
2	Новий продукт	Вихід на ринок. Скорочення монополії. Надання нових рішень в області	Розробка нових функціональ-них можливостей. Нові продукти, що надходять на ринок. Надання різних типів ліцензій в залежності від

			потреб користувача\замовника.

Продовження таблиці 8.7

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
3	Вихід аналогу	Надання продукту з певними характеристиками й можливостями, недоступними у конкурентів	Аналіз ринку і споживача для того щоб відповідати їх потребам і забезпечити функціональність в якнайкорот-шому часу для ціни яка дешевше ніж ця з замінюючих продуктів
4	Конкуренція	Зниження довіри до конкурента 1 внаслідок помилок в функціонуванні	Акцентувати увагу на надійності системи
5	Держава	Послаблення обмежень в законодавстві	Оптимізація діяльності для скорочення витрат

За результатами аналізу пропозиції описуваного продукту, наведеного в таблиці 8.8 і необхідного для виходу на ринок, описаного в таблиці 8.4, визначено загальні риси конкуренції на ринку стартап-проекту і зазначено можливий вплив на діяльність підприємства, можливі дії й діяльність компанії, спрямовані на підвищення її конкурентоспроможності.

Таблиця 8.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: - монополія	Товари з кожної компанії на ринку, є недосконалим замінником товарів, що реалізуються іншими компаніями	Створення продукту з характеристиками, що охоплюють область застосування, не поширюється на інші товари-замінники
2. За рівнем конкурентної боротьби - національний	Всі замінники продуктів були розроблені міжнародними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Доступ до ринку продажі продукції з необхідним клієнтом функціональністю. Створення маркетингу на головних інтернет-ресурсах для охоплення великої чисельності

		потенційних користувачів

Продовження таблиці 8.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3. За галузевою ознакою - внутрішньогалузева	Цей тип продукту можна застосовувати в різноманітних галузях	Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу. Підтримування всім відомих способів взаємодії з середовищем проектування
4. Конкуренція за видами товарів: - товарно-видова	Ця конкуренція – конкуренція між товарами одного роду	Запровадження функціональності, яка не доступна в товарів-замінників. Полегшення інтерфейсів. Надання підтримки
5. За характером конкурентних переваг - цінова / нецінова	Цінові переваги – точка комерціалізації. Не цінове забезпечення функціональності, яка недоступна в товарах-замінниках	Забезпечення платних ліцензій лише на критично важливі функції для клієнта з певним періодом підтримки, як вказано у відповідній ліцензії. Реалізація унікального функціоналу
6. За інтенсивністю - марочна	Наявність оригінальної марки, що відрізняє цей продукт від продуктів-замінників	Впровадження власної назви та власного знаку

Як зазначалося вище, конкуренція відіграє виключно важливу роль у розвитку компанії. У зв'язку з цим в таблиці 8.9 представлений аналіз конкуренції в галузі за кількома її компонентами, по кожному з яких зроблені висновки.

За результатами аналізу пропозиції описуваного продукту, наведеного в таблиці 8.9 необхідного для виходу на ринок, описаного в таблиці 8.4, визначені основні особливості конкуренції на ринку стартап-проекту і наведені висновки по кожному з компонентів аналізу для успішної конкуренції на ринку.

Таблиця 8.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Існують 3 конкуренти на ринку	Відсутні	Відсутні	Майже всі галузі	Системи з аналогічною функціональністю, але меншою вартістю
Висновки:	Прямі конкуренти намагаються сконцентруватися на інших напрямках свої продуктів	Загроза появи нових конкурентів, але конкуренти можуть не мати досвіду в реалізації масштабування проекту	—	Клієнтам необхідно забезпечити наявність співвідношення ціни / якості на ринку	Загроза втрати позицій системи на ринку, з таким же функціоналом, але меншою вартістю

Вивчивши можливості роботи на ринку з огляду на конкурентну ситуацію, можна зробити висновок: оскільки кожен з існуючих продуктів в значній мірі не впливає на поточну ситуацію на ринку в цілому, кожен з існуючих продуктів має свій конкретний обсяг і свої позитивні й негативні сторони, щодо вирішення певних видів завдань, вихід на цей ринок є можливою і реалізованим завданням.

Для виходу на ринок продукт повинен містити певну функціональність, яка відсутня в аналогових продуктах, має задовольняти потреби користувачів, мати необхідний і достатній функціонал з конфігурацією, підтримкою з боку розробників і можливістю розробки спеціального функціоналу під відповідну ліцензію.

Таблиця 8.10 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Швидкість роботи	Бистрота роботи відіграє велику роль для користувачів, так як вони не будуть готові чекати кілька хвилин для отримання

		результату роботи додатку

Продовження таблиці 8.10

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
2	Прагматичність	Після запуску стартапу система не буде дуже складною з точки зору архітектури на перший погляд. Після певного періоду з додаванням функціоналу і оптимізацією алгоритмів, програмний код буде складнішим. Цей етап настане не раніше, ніж через рік постійної роботи над проектом
3	Зручність	Інтерфейс користувача перш за все це комфорт при використанні, розташування елементів управління повинно бути інтуїтивно зрозуміло, зручність використання системи зіграє важливу роль у можливості конкурувати з іншими гравцями ринку
4	Оптимізація	Якщо додаток виходить з ладу занадто часто, користувачі не будуть вважати додаток надійним
5	Налаштування під користувача	Одні люди мають різні звички, які вони використовують, наприклад, якщо є люди, які люблять працювати з додатком, де є темні кольори, і є люди, які люблять світлі кольори. Можливість редагування зовнішнього вигляду програми дає значну перевагу серед конкурентів
6	Відкритість вихідного коду	Будь-який продукт при наявності вихідного коду має перспективи розвитку в багатьох напрямках, особливо тих, які на перший погляд можуть бути не очевидні
7	Приватність	В останні роки приватним життям та інформацією людей все частіше зловживають шахраї або великі корпорації, яким необхідно узгодити умови доступу до приватної інформації та її обробки
8	Технічна підтримка	Якщо технічна підтримка компанії буде працювати своєчасно й оперативно, це

		допоможе зберегти репутацію компанії на відміну від конкурентів, де на неї не звертають уваги

Продовження таблиці 8.10

9	Документація	Усякий засіб, особливо якщо він має новий функціонал, має бути добре пояснено своїм користувачам і як його можна використовувати, щоб не виникало з ним проблем з подальшою роботою
10	Ціна	Чим дешевше продукт, тим більше шансів, що його можливо куплять, особливо якщо він працює краще, ніж конкурентоспроможний продукт

Згідно з аналізом факторів конкурентоспроможності продукту необхідних для виходу на ринок, описаних в таблиці 8.4, основні фактори конкурентоспроможності, наведені в таблиці 8.10.

На основі аналізу перерахованих вище факторів конкурентоспроможності проекту, аналіз її сильних і слабких сторін, результат якого представлений в таблиці 8.11 для оцінки конкурентоспроможності за 20-бальною шкалою.

Таблиця 8.11 — Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			–3	–2	–1	0	+1	+2	+3
1	Прагматичність	17		+					
2	Зручність	9					+		
3	Швидкість роботи	5						+	
4	Оптимізація	5						+	
5	Налаштування під користувача	5						+	
6	Відкритість вихідного коду	12				+			
7	Приватність	12				+			
8	Технічна підтримка	5						+	
9	Документація	5						+	

10	Ціна	12				+			

За результатами аналізу сильних і слабких сторін проекту, наведеного в таблиці 8.10 і необхідного для виходу на ринок, описаного в таблиці 8.4, було визначено, що найбільшою перевагою стартап-проекту є його прагматичність, наявність відкритого вихідного коду, приватність та ціна. Результати аналізу, наведеними в таблиці 8.11, будуть використовуються для SWOT-аналізу, який є завершальним етапом ринкового аналізу можливостей реалізації проекту.

Таблиця 8.12 — SWOT- аналіз стартап-проекту

Сильні сторони: Прагматичність системи через її легкість роботи; Простота у використанні; Наявність відкритого вихідного коду;	Слабкі сторони: Неоптимізованість алгоритму; Швидкість роботи системи;
Можливості: Зворотній зв’язок з клієнтурою компанії для спроможності розвивати проект в інші напрямки.	Загрози: Складність роботи алгоритму

На основі SWOT-аналізу розроблені альтернативи ринкової поведінки для виводу стартап-проекту на ринок і зразкові оптимальні строки їх реалізації на ринку з урахуванням потенційних проектів конкурентів, які можуть бути виведені на ринок. Для деяких альтернатив був проведений аналіз з точки зору термінів і ймовірності отримання ресурсів, результати якого наведені в таблиці 8.13.

Таблиця 8.13 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певних можливостей для використання споживачами протягом обмеженого періоду часу	70 %	2-3 місяці
2	Реклама	75 %	1-2 місяці
3	Написання статей та опис товару на відомих	50 %	2-3 тижні

	ресурсах		

Продовження таблиці 8.13

4	Презентація товару на хакатонах й інших ІТ заходах	60 %	1-3 місяці

8.4 Розроблення ринкової стратегії проекту

Перший крок у розробці ринкової стратегії припускає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 8.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Будівельники які виконують роботи на зовні	Присутня	Низький	Присутня	Важка
2	Судноплавники які часто займаються перевезеннями	Присутня	Середній	Присутня	Середня
3	Авіаційні компанії для розробити перелік попереджувальних дій	Присутня	Середній	Присутня	Важка
4	Аграрники які закладують початок посівних робіт	Присутня	Середній	Присутня	Середня
Які цільові групи обрано:2, 4					

Підсумок розгляду цільових груп потенційних споживачів стартап-проекту наведені в таблиці 8.14, дозволяють визначити, якими перевагами продукту є

можливість використання для виходу в даний сегмент ринку, а також доцільно використовувати ресурси для впливу на певну групу. За результатами аналізу можна зробити висновок, що відповідною цільовою групою для поширення даного програмного продукту є сільськогосподарські робітники й суднобудівники. У відповідності зі стратегією охоплення ринку товарами була обрана стратегія масового маркетингу, оскільки масову аудиторію в цілому забезпечує стандартизований продукт з можливістю розширення функціоналу за домовленістю (згідно з ліцензією).

Таблиця 8.15 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами. Формування лояльності і прихильності споживачів	Зниження ступеню замінності товару. Прихильність клієнтів. Відмітні властивості товару. Відмітні характеристики товару	Стратегія диференціації

За результатами визначення базової стратегії розвитку, представленої в Табл. 8.15, була обрана оптимальна стратегія розвитку для альтернативи, обраної в Табл. 8.13, повинна відповідати основним потенційним групам споживачів стартап-проекту, розглянутих в таблиці 8.14.

За результатами, отриманими в ході вибору базової стратегії розвитку стартап-проекту, яка включає в себе такі компоненти, як альтернативне розвиток, стратегія покриття ринку, конкурентна позиція, була обрана диференціальна базова стратегія.

При цьому необхідно також вибрати стратегію конкурентної поведінки, представлену в таблиці 8.16.

Таблиця 8.16 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні, тому що є замінники продуктів, але ці замінники продуктів не мають необхідної функціональності	Таким чином, метою компанії є пошук нових клієнтів і, зокрема, усунення існуючих конкурентів для задоволення потреб останніх	Компанія трохи копіює особливості продукту конкурента, основною метою компанії є розробка нового унікального функціоналу, з підтримкою головного функціоналу конкурентів	Стратегія заняття конкурентної ніші

За результатами визначення основної конкурентної поведінки, представленої в таблиці 8.16, була обрана оптимальна стратегія конкурентної поведінки з використанням основних характеристик першого кроку конкурентів, описаного в таблиці 8.2, для якого наявність пошуку споживачів і заняття конкурентної ніші як стратегії безпосередньо, має забезпечити можливість ефективної боротьби з вищезазначеними конкурентами.

Таблиця 8.17 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Адаптивність, простота в	Диференціація	Адаптивність, кро-сплатформність,	Адаптивність Конфіденційність

	налаштуванні, конфіденційність		простота в налаш-туванні	Простота

За підсумками визначення стратегії позиціювання, представленої в таблиці 8.17, була обрана оптимальна стратегія позиціювання стартап-проекту з урахуванням головних вимог до продукту в цільовій аудиторії. З головних конкурентних позицій проекту, представлені в таблиці 8.15, були дві асоціації обрано відповідає вимогам: гнучкість і простота. Третя позиція для асоціацій була неконкурентоспроможною, але головна вимога до будь-яких проектів у цій галузі — конфіденційність.

8.5 Розроблення маркетингової програми стартап-проекту

Перший крок — формування маркетингової концепції продукту, який одержить споживач. Для цього в таблиці 8.18 слід узагальнити підсумки попереднього аналізу конкурентоспроможності товарів.

Таблиця 8.18 — Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Прогнозуваннння погоди на довгостроковий період	Високоточний прогноз погоди на окремий участок на один місяць вперед	Прогнозування погоди
2	Адаптивність	Включення нових модулів в систему без зайвих зусиль	Слабка зв’язаність компонентів

На основі результатів визначення ключових переваг концепції потенційного продукту, представлених у таблиці 8.18, розглянуто основні потреби ринку та проведена оцінка переваг проекту. Визначено ключові переваги перед конкурентами, що задовольняють заданим базовим потребам. Такі ключові переваги полягають у використанні високоточного прогнозування і слабкої зв'язності компонентів.

Таблиця 8.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Чисельна модель прогнозування погоди		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	Зручність	Нм	Е
	Швидкість роботи	Нм	Тх
	Оптимізація	Нм	Тх
	Приватність	Нм	Тх
	Налаштування під користувача	Нм	Ор
	Технічна підтримка	Нм	Тх
	Документація	Нм	Тл
	Ціна	Нм	Е
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
III. Товар із підкріпленням	До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товар		
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент			

За підсумками трирівневої маркетингової моделі продукту, представленої в таблиці 8.19, визначено уявлення про продукт, його властивості й характеристики. Визначався засобами захисту від копіювання: патентним правом та електронними ключами на фізичному рівні.

Таблиця 8.20 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники, грн	Рівень цін на товари-аналоги, грн	Рівень доходів цільової групи споживачів, грн	Верхня та нижня межі встановлення ціни на товар/послугу, грн
1.	33000	45000	260000	27000-40000

Таблиця 8.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Придбання місячної, квартальної, піврічної або річної підписки	Продаж	0 (напряму) 1 (через одного посередника)	Власна та через посередників

За підсумковими дослідженнями меж цін і оптимальної системи продажів представлений в таблицях 8.20 і 8.21, з урахуванням цін аналогічних і взаємозамінних товарів, верхній і нижній межі цін були встановлені. Крім того, була сформована система продажів, що передбачає продовження ліцензії шляхом придбання періодичної підписки: щомісячної, квартальної, піврічної або річної підписки та описує канали продажів, в результаті чого система продажів визначалася як власна, так і через посередників.

В якості завершуючого компонента маркетингової програми була розроблена концепція маркетингових комунікацій, для якої були використані результати попереднього вибору бази позиціювання і певні особливості поведінки клієнта, результати яких наведені в таблиці 8.22.

Таблиця 8.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Придбання ліцензії шляхом отримання унікального ключа	Інтернет	Адаптивність Конфіденційність Простота	Показати перспективу користування системою, у тому числі і перед конкурентами	Демо-ролик застосування та таргетингова реклама

За результатами розробки концепції маркетингових комунікацій, представленої в таблиці 8.22, з використанням результатів формування системи продажів,

наведених у таблиці 8.21, була визначена специфіка поведінки цільових клієнтів; інтернет як основний канал комунікації цільових клієнтів; адаптивність, конфіденційність, простота в якості ключових позицій для позиціонування; а також сформовано завдання рекламних повідомлень, концепція рекламної обробки.

8.6 Висновки

Відповідно до пройденого дослідження, є наявною можливість комерціалізації проекту на ринку. Варто зазначити, що є перспективи здійснення впровадження, враховуючи потенційні клієнтські групи, невисокі вхідні бар'єри, і проект має дві істотні переваги перед конкурентами.

В процесі підготовки документації було сформовано ідею стартап-проекту, приведено порівняння з низкою потенційних конкурентів, що дало спроможність оцінити конкурентоспроможності та можливість і складність стартапу ввійти в ринок. Підсумки огляду головних технологічних стеків, які можуть бути застосовані для реалізації стартап проекту та вибраний оптимальний стек. Були представлені результати аналізу характеристик попиту на ринку.Зроблений аналіз цільових аудиторій користувачів представленого продукту, аналіз ймовірних факторів, загроз й аналіз можливих сприятливих умов для стартап-проекту.

Проведено дослідження чинників конкурентоспроможності та зіставлення аналізу сильних й слабких сторін проекту, підсумки якого згодом були застосовані для SWOT-аналізу стартап-проекту.

Були вибрані цільові групи потенційних споживачів, визначена головна стратегія розвитку й стратегія конкурентної поведінки та позиціювання, розроблена маркетингова програма.

У результаті був зроблений вивід, що для процвітання реалізації проекту необхідно розрахувати головні фінансово-економічні показники проекту, а також управління потенційними ризиками. Проаналізувавши здобуті результати, можна зробити вивід про доцільність подальшої реалізації.

ВИСНОВКИ

В ході магістерської дисертації було здійснено автоматизоване проектування високорівневих алгебро-алгоритмічних специфікацій програмного забезпечення для вирішення задачі метеорологічного прогнозування. Прогнозування відбувається на базі паралельної реалізації завдань моделювання атмосферної циркуляції на багатопроцесорній платформі. На фундаменті застосування спроектованих засобів автоматизованого проектування та синтезу програм, виконана генерація програмного коду мовою програмування C++ за побудованими технічними умовами. Перевагою підходу до програмного проектування використовуваного в інструментах, є застосування мовних структур, які є дуже близькі до природної мови, а також використання методу, що забезпечує синтаксичну точність розроблюваних алгоритмів й програм.

Проведений експеримент з виконанням реалізації згенерованого паралельного застосування для вирішення задачі метеорологічного прогнозування на багатоядерній платформі Intel Xeon Phi, дуже хороші результати якого продемонстрували високу швидкість багатопроцесорного прискорення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Б. Страуструп. Язык программирования C++ = The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с.

2. А.Ю. Дорошенко, П.А. Іваненко, О.М. Овдій, О.А. Яценко Автоматизоване проектування програм для розв’язання задачі метеорологічного прогнозування

3. Ф.И. Андон, А.Е. Дорошенко, К.А. Жереб, Р.С. Шевченко, Е.А. Яценко, Методы алгебраического программирования. Формальные методы разработки параллельных программ. - Киев, "Наукова думка".-2017.- 440 с.

4. OpenMP [Електрон- ний ресурс]. – Режим доступу: http://openmp.org

5. 2016ВКР230628УЗОМА.PDF [Електронний ресурс]. – Режим доступу: http://library.eltech.ru/files/vkr/bakalavri/2306/2016%D0%92%D0%9A%D0%A0230628%D0%A3%D0%97%D0%9E%D0%9C%D0%90.PDF

6. Процессор Intel Xeon Phi 7210 94033 [Електрон- ний ресурс]. – Режим доступу: https://www.intel.ru/content/www/ru/ru/products/processors/xeon-phi/xeon-phi-processors/7210.html

7. Обзор Intel Xeon Phi [Електронний ресурс]. – Режим доступу: http://www.thg.ru/cpu/obzor_intel_xeon_phi/index.html

8. Intel Delivers New Architecture for Discovery with Intel Xeon Phi Coprocessors | Intel Newsroom [Електронний ресурс]. – Режим доступу: https://newsroom.intel.com/news-releases/intel-delivers-new-architecture-for-discovery-with-intel-xeon-phi-coprocessors/

9. Предмет, задачи и направления метеорологии [Електронний ресурс]. – Режим доступу: http://yznaika.com/notes/451-meteorologia

10. Про використання деяких даних Глобальної Системи Прогнозування [Електронний ресурс]. – Режим доступу: http://accuweather.org.ua/gfs025/

11. Дорошенко А.Ю., Томишин Ю.В., Яценко О.А Проектування програми метеорологічного прогнозування для багатоядерної платформи – 2018

12. Андон Ф.И., Дорошенко А.Е., Яценко Е.А. и др. Алгеброалгоритмические модели и ме- тоды параллельного программирования. – Киев: Академпериодика, 2007. – 631 с.

13. Дорошенко Е.А., Яценко Е.А. О синтезе программ на языке Java по алгеброалгори- тмическим спецификациям // Проблеми програмування. – 2006. – № 4. – С. 58–70.

14. Дорошенко А.Е., Жереб К.А., Яценко Е.А. Формализованное проектирование эффективных многопоточных программ // Там само. – 2007. – № 1. – С. 17–30.

15. Яценко Е.А. Интеграция инструментальных средств алгебры алгоритмов и переписывания термов для разработки эффективных параллельных программ // Там само. – 2013. – № 2. – С. 62–70.

16. Дорошенко А.Ю., Бекетов О.Г., Яценко О.А. та інші. Розробка сервісно-орієнтованих засобів для запуску паралельних програм на мультипроцесорному кластері // Там само. – 2014. – № 4. – С. 3–14.

17. Иовчев В.А., Мохница А.С. Инструмента- льные средства алгебры алгоритмики на платформе Web 2.0 // Там само. – 2010. – № 2–3. – С. 547–555.

18. Дорошенко А.Ю., Бекетов О.Г., Яценко О.А. та інші. Автоматизована генерація паралельних програм для графічних прискорювачів на основі схем алгоритмів // Там само. – 2015. – № 1. – С. 19–28.

19. Андон Ф.И., Дорошенко А.Е., Яценко Е.А. и др. Инструментальные средства автоматизации параллельного программирования на основе алгебры алгоритмов // Кибернетика и системный анализ. – 2015. – № 1. – С. 162–170.

20. Doroshenko A., Shevchenko R. A rewriting framework for rule-based programming dynamic applications // Fundamenta Informaticae. – Amsterdam: IOS Press, 2006. – Vol. 72, N 1–3. – P. 95–108.

21. Черниш Р.І. Паралельна реалізація моделі макромасштабної циркуляції атмосфери // Вісник Київського національного університету імені Тараса Шевченка: серія фізико-математичні науки. – 2009. – № 2. – С. 155–158.

22. Черниш Р.І., Тирчак Ю.М., Іваненко П.А. Побудова паралельного алгоритму чисельного розв’язання багатовимірної задачі моделювання навколишнього середовища // Проблеми програмування. – 2009. – № 1. – С. 85–91.

23. Прусов В.А., Дорошенко А.Е., Черныш Р.И. Метод численного решения многомерной задачи конвективной диффузии // Кибернетика и системный анализ. – 2009. – № 1. – С. 100–107.

24. OpenMP Application Program Interface [Електронний ресурс]. – Режим доступу: http://www.openmp.org/mp-documents/ OpenMP4.0.0.pdf

25. Sannella D., Tarlecki A. Foundations of algebraic specification and formal software development. – Berlin: Springer-Verlag, 2012. – 594 p.

26. Flener P. Achievements and prospects of program synthesis // Lecture Notes in Artificial Intelligence. – Berlin: Springer- Verlag, 2002. – Vol. 2407. – P. 310–346.

27. Gulwani S. Dimensions in program synthesis // Proc. 12th Int. ACM SIGPLAN symposium on Principles and Practice of Declarative Programming, Hagenberg, Austria (26–28 July, 2010). – New York: ACM, 2010. – P. 13–24.

28. Raghesh A. A framework for automatic OpenMP code generation. A project report [Електронний ресурс]. – Режим доступу: http://www. cse.iitm.ac.in/~raghesh/raghesh- a-masters-thesis.pdf

29. Hu K., Zhang T., Yang Z. Multi-threaded code generation from Signal program to OpenMP // Frontiers of Computer Science. – Berlin: Springer-Verlag, 2013. – Vol. 7, N 5. – P. 617–626.

30. PLUTO – an automatic parallelizer and locality optimizer for multicores [Електрон- ний ресурс]. – Режим доступу: http://pluto- compiler.sourceforge.net/